2011.09

PROGRAMMER 程序员

敏捷！

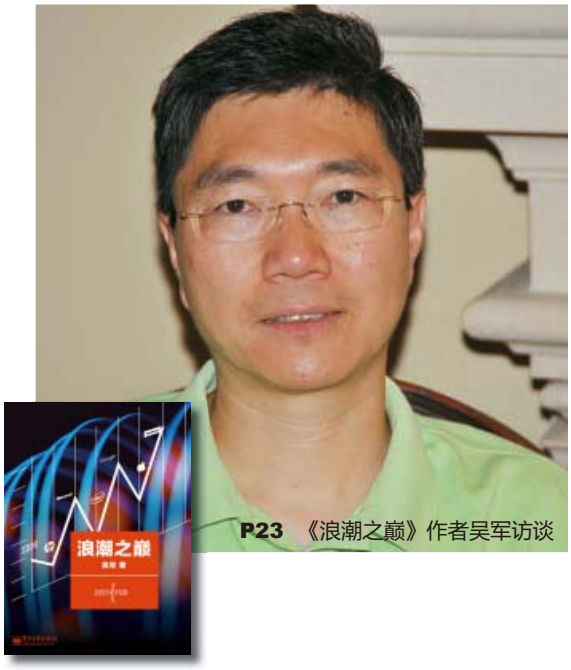
ISSN 1672-3252

9 771672 325111 09>

邮发代号：2-665　定价：15元

www.programmer.com.cn　www.csdn.net

Contents



P23 《浪潮之巅》作者吴军访谈

P28

资讯

热点报道

封面报道：敏捷！

敏捷并不完美，一直在演进。对于敏捷我们没有现成的答案。
敏捷不是银弹，只是接受常识。单纯地依靠任何一种敏捷技术都不可能解决问题。
敏捷不反对方法学，相反努力为之正名。鼓吹敏捷以“传统”为敌，并不是真正拥护敏捷的人。
在本期特别策划中，我们邀请了不同开发理念的拥护者，以PMP、CMMI等不同视角，探问敏捷的得失利弊；以UX、Tools、Testing等不同维度，探问敏捷的发展与演进。在反复探问与比较中，虽然我们无法抵达问题的终点，但或许能找到一个适合自己的起点。

管理

Contents

P100

P128

2011年9月刊　总第227期

Contents

主管：中国社会科学院
主办：中国社会科学院文献信息中心
出版：《程序员》杂志社
网址：http://www.programmer.com.cn
国际刊号：ISSN 1672-3252
国内刊号：CN11-5038/G2
邮发代号：2-665
广告经营许可证号：京东工商广字0188号

总编：黄长著　Editor-in-chief：Huang Changzhu
社长/常务副总编：张悦校　President：Zhang Yuexiao
副社长：蒋 涛　Vice President：Jiang Tao
编委会：黄长著　张悦校　陈洋彬　蒋涛　曾登高　刘江
Editorial Member：Huang Changzhu　Zhang Yuexiao　Chen Yangbin
Jiang Tao　Zeng Denggao　Liu Jiang

执行主编：孟迎霞　Executive Editor-in-chief：Meng Yingxia
编辑部主任：常政　Director：Chang Zheng
责任编辑：董世晓　高松　陈博
Editors：Dong Shixiao　Gao Song　Chen Bo
特邀编辑：方梁　高昂　赵健平　吕娜　卢鹄翔
Contributing Editors：Fang Liang　Gao Aang　Zhao Jianping
Lv Na　Lu Dongxiang
美术设计：纪明超　Art Designer：Ji Mingchao
美术编辑：林象海　Art Editor：Lin Xianghai
流程编辑：白羽中　Coordinator：Bai Yuzhong
Tel：010-64351458
E-mail：editor@csdn.net

广告总代理：北京创新乐知信息技术有限公司
Sole Advertising Agency：Bejing CSDN Co.,Ltd
Tel：010-64376055
E-mail：ad@csdn.net
Marketing Dept：010-51661202（ext 149）
E-mail：market@csdn.net

发行部
Distribution Dept.010-64351431
E-mail：sales@csdn.net

读者服务部
Readers service Dept.
网上订购：http://dingyue.programmer.com.cn/
读者信箱：reader@csdn.net
地址：北京市朝阳区广顺北大街33号院1号楼福码大厦B座12层
Address: B-12th Floor Fairmont Tower NO.33 Guangshun North street,Chaoyang District,Beijing
邮政编码：100102
电话：010-64351436
传真：010-64348545

法律顾问：北京中润律师事务所 王杰
Law Consultant：Beijng Hengsheng Lawyer Firm
印刷：北京盛通印刷股份有限公司
Print: Beijing Shengtong Printing Co., Ltd.
出版日期：每月1日
Publication Date：the first day per month
零售价：RMB 15.00元　新台币 390元　HK $ 35.00（港、澳）
US $ 9.00（海外）
Retail Price：RMB 15, NT$390, HK $ 35.00,US $ 9.00

当敏捷成为主流

2001 年 2 月，“ 敏捷宣言”在美国犹他州雪鸟度假村诞生，至今已超过十年。而敏捷也如野火燎原，似乎真正成为主流。在欧美国家，从数人的小团队到超过万人的大型团队，众多软件研发机构都不同程度地采纳了敏捷实践。支持敏捷也成了微软、IBM等软件开发工具厂商宣传产品必备的亮点。Alistair Cockburn更举了一个直观的例子，犹他州已经规定医院的IT项目必须采用敏捷开发。

与此同时，敏捷也从软件开发向外拓展，通过DevOps和持续交付开始影响运维社区，通过敏捷UX进入用户体验和产品设计界，甚至非IT的通用产品研发的敏捷转型也成为一种现象。2005年敏捷项目独立宣言发布，标志着敏捷更进入到企业管理者的视野，开始影响组织转型、企业管理和文化变革。

敏捷成为主流的最明显的标志，是与早些年相比，已经很少有人会强调自己的开发流程不是敏捷，公开反对敏捷也成了不太合时宜的事情。SD Times杂志今年三月甚至放出了“敏捷已死”的标题党文章，理由是现在既然没有非敏捷的项目和团队，这个术语也没有再存在的必要……

敏捷运动真的已经大功告成了吗？情况当然没有那么乐观。

事实上，经常被引用为敏捷已成主流依据的2010年Forrester报告中，采用敏捷过程的团队占比也只有35%。而且敏捷项目实施的失败率也很高，Ken Schwaber估计尝试Scrum的组织有75%未达预期。在中国，今年上半年CSDN和本刊联合举办的“中国软件开发者年度调查”中，选择“没有特别的开发过程”的占29.3%，选择“自制过程”的23.9%，选择CMMI的有9.9%，合计63.1%。相比而言，选择Scrum的只有7.2%，有29.7%选择极限编程。由于国内的文化传统和技术管理水平都与敏捷的土壤相去较远，虽然已经取得了可喜的进步，但要走的路显然还很长。

值得警觉的是，近来国内外敏捷社区中都出现了一些僵化和脱离实际的迹象。敏捷的主流本来起源于草根程序员们的实践，本质上是对自上而下的命令控制式传统官僚管理模式的革命，它的思想来源非常丰富，社区充满生机，而且脚踏实地、实事求是。但是十年后，最容易入门、更多只是一个项目管理框架的Scrum成为敏捷的代名词，Scrum基金会推出一系列违反敏捷精神的认证培训项目，没有多少项目经验、编程经验的专业敏捷专家（经过简单的培训就能获得Master的称号）越来越多，各路培训机构纷纷穿上敏捷外衣，敏捷也开始被教条化、神圣化，空谈而无视客户价值……在我看来，这些都不是什么好兆头。

十年重聚的时候，多位敏捷宣言创始人都表示宣言本身仍然有效，无需修改。平心而论，敏捷宣言的理念某种意义上的确具有永恒的价值。但是这种态度还是隐含着危机。我们都知道，敏捷诞生于1990年代初期，主要源自企业级软件开发。虽然其发展与互联网第一次浪潮高歌猛进有关，但总体而言敏捷方法并不完全兼容互联网时代。2000年后兴起的以Google和Facebook为代表的互联网公司，他们研发的敏捷程度、迭代周期恐怕都超出敏捷大师们当年所能想象。换句话说，他们比敏捷更敏捷。这里面难道没有值得学习的地方？我们都知道诺基亚可能是Scrum Master最多的公司。

此外，敏捷先驱们基本上是清一色的程序员、技术经理，没有用户体验人员，也缺乏其他背景，因此宣言里的一些用语对于今天敏捷无处不在的形势也不够用了。比如，“可工作的软件”里光可以工作显然不够了，还要能够打动用户（想想苹果怎么做的），软件也太局限了，应该改为解决方案或者产品。Scott Ambler还建议将“客户”改为项目干系人。

还有更重要的，敏捷的核心是人和变化。研究人需要心理学和脑科学方面的艰苦探索；拥抱变化更是这世间最困难的事情。

当敏捷成为主流，同志仍需努力。

邮件：liujiang@csdn.net
新浪微博：@刘江CE
欢迎大家交流反馈，欢迎投稿、批评、建议和挑错

webOS被放弃

8月19日凌晨，惠普在发布第三季度财报的同时，也发布了惊人的消息：将以100亿美元左右的价格收购海量数据公司Autonomy，同时计划分拆PC业务，其中的webOS手机和平板电脑业务将被放弃。webOS是基于Linux的一个专有移动操作系统，2009年由Palm推出，纳入Web 2.0、云同步、多任务和开放架构等新元素，内置Mojo Web框架，应用开发模型很好地融合了Native和Web方式，技术上很有独到之处。惠普于2010年4月以12亿美元收购了Palm，获得了该公司的硬件业务，以及围绕webOS的软件业务。2011年2月，惠普宣布推出TouchPad平板电脑。这款平板电脑由惠普设计，采用Palm webOS 3.0系统。这款平板电脑原计划于今年7月面市，惠普还将于8月推出Pre 3智能手机。2011年5月，惠普宣布推出收购Palm之后的首款智能手机Veer。推出首月，TouchPad平板电脑的销量只有2.5万台。尽管获得了不错的评测，但消费者仍很少购买这款产品。惠普随后被迫下调TouchPad的价格，并面向Palm Pre的用户提供折扣。目前在全美的百思买门店中，有20万台TouchPad积压。

土豆网上市

土豆网于美国当地时间8月17日周三成功登陆纳斯达克市场，开盘价25.11美元，较发行价29美元下跌13.41%。截至收盘，土豆网报价25.56美元，较发行价下跌11.86%。盘中该股成交价最低为23.50美元，最高为27.75美元。此次土豆上市发行600万股ADS（美国存托凭证），每股美国存托凭证相当于4股B类普通股。其中557 万股ADS为新发行股票，43万股ADS为土豆网创始人和CEO王微个人抛售。此次土豆上市募集资金总额为1.74亿美元，瑞士信贷和德意志银行将出任土豆网此次首次公开招股的联席承销商。据土豆网向美国证券交易委员会（SEC）提交的招股说明书透露，土豆在IPO上市后，王微将抛售43万股ADS。王微的持股将由原来12.7%降至IPO之后的8.6%。受土豆网IPO影响，同在美上市的视频网站优酷和酷6分别上涨12.58%和10.31%，盘中两支个股涨幅一度双双超过20%。

小米手机发布

8月16日，小米科技在北京发布小米手机。据小米科技CEO雷军介绍，小米手机是国内首款双核1.5GHz主频手机，为全球主频最快智能手机，小米手机只在小米网上零售，9月5日上线接受预订，预计10月初量产销售，售价1999元。

小米手机采用的MIUI系统，由Android 2.3.5版本定制修改而来。雷军称MIUI首创用互联网模式开发手机OS的先例，50万发烧友参与开发改进（三分之一的改进创意来自网友），每周迭代升级一次。雷军说，硬件配置是小米手机的强项，可以超过目前市面上所有在售的智能手机。雷军还宣布，小米手机将开放刷机设置，官网提供MIUI和Android原生系统，未来MIUI还将支持更多品牌的手机。

iAd前景堪忧

Apple负责移动广告的副总裁Andy Miller计划辞职，加盟投资公司Highland Capital，所负责的业务暂时与移动无关。Andy Miller于2006年与人共同创办了移动广告公司Quattro Wireless，2010年初以2.75亿美元被Apple收购，Miller也随之成为IT巨头的副总裁，直接向乔布斯报告。当时，乔布斯曾说过：“我们本来想买AdMob的，但Google插进来抢走了。所以我们买了Quattro，他们正在教我们怎么做广告，与我之前所见都不一样。”Quattro随后被关闭，改造后以iAd平台的形式推出，支持在iOS设备上的应用内提供广告。但与Google AdMob、Facebook和其他创业公司增长迅猛相比，Apple的移动广告业务一直未见起色。广告主一直抱怨Apple的控制太多，而且价码也偏高（最低投放额是100万美元，每次用户浏览1美分，每次点击2美元）。最近情况有所改变（最低金额降到了50万美元），但Miller的离职说明这块业务应该无法令乔布斯满意。AppAdvice报道，iAd在2010年6月推出后，到2011年5月广告主才突破100家。移动广告是一个飞速增长的领域，《华尔街日报》文章曾指出，2010年美国移动广告从一年前的4.16亿美元增长到7.43亿美元，增速达到79%。

“如果我是Android硬件生产商、Android手机运营商或者与Android手机休戚相关的任何第三方，将会拿起自己的Android手机拨通某位Google高管的电话质问‘我预见到了未来的威胁迹象’。”

——Stephen Elop在赫尔辛基出席一个论坛时如此表示。随着Google宣布斥资125亿美元收购摩托罗拉移动，业内分析师纷纷对三星和HTC将处于不利位置表示担忧。Stephen Elop也表示，这些担忧很可能变为现实。

“奇虎360有能力继续保持在网络安全领域的领先和权威地位。”

——周鸿祎对奇虎360最新财报的发言。2011财年第二季度未经审计财报显示，该公司第二季度营收为3510万美元，创下历史最高记录，比去年同期的1270万美元增长176.6%；净利润为1110万美元，也创下历史最高记录，比去年同期的220万美元增长411.5%。

“我们正向着更加网络化的做法迈进，用户使用的版本号并不重要，只要他们用的是最新版本。我们目标是让用户摆脱版本号的束缚。”

——Mozilla计划取消Firefox“关于”对话框中版本号信息。Mozilla社区主管Asa Dotzler对这一举动做出了解释，不过大部分用户并不认同这种做法。

“百度：阳光背后的阴影”

——CCTV2财经频道“经济与法”栏目以此为题的报道。8月15日以来，百度再次遭到中央电视台多套栏目“曝光”，更被指过滤虚假网站不力、收受虚假网站更高的费用以致网民受骗上当。这已是百度近年来第三次遭到央视的类似指责。

“Windows操作系统面临竞争，主要来自于苹果和Google。”

——微软向SEC提交了2011年年报的10-K表格，相比于2010年，明显的修改是忽略了Linux，不再把它当作桌面对手。

“此次的Nate案件明显地暴露出因实名制而收集个人信息并保管的风险。”

——韩国三大门户网站之一Nate和社交网站赛我网遭到黑客攻击，约3500万名用户的信息外泄，这使得韩国实行的“互联网实名制”协议饱受指责。韩国国会立法调查处发表的《Nate遭黑客攻击和门户网站的个人信息保护》报告称此协议远未达成预期效果。

CSDN十大新闻

2011年7月26日~8月26日

01 未来IT行业将缩减到三类职业

作者 / Jason Hiner

IT领域发生着巨大的改变，这些改变产生的焦虑把人的注意力转移到三大类IT工作上，分别是：IT顾问、项目管理者和开发者。做出这样论断的主要原因是，IT环境彻底改变，越来越多的传统软件迁移到Web上，用户也不像以前那样需要更多的帮助。

02 李彦宏15年前专利曝光

作者 / 乐天

本文讲述了百度的发家史。1996年，李彦宏提出“超链分析”概念，并于1997年2月申请了专利：“超链分析技术”，正是它，催生并且支撑起了目前百度庞大的市值。虽然外界多有争议，但李彦宏认为，对搜索技术的专注和创新，才是百度成长的关键。

03 防止IT技术人员被挖走的五大措施

作者 / Pam Baker

如何挽留住核心IT技术人员，是每位领导者都在探索的问题。本文指出要重建公司的提升与惩罚机制，可以由以下措施挽留IT技术人员：让他们入股公司；提供一种生活，而不仅仅是一份工作；提供学习的机会；让他们去完成；让他们去创造。

04 17岁天才少年进入TechStars创业

作者 / Alyson Shontell

互联网营销专家Seth Godin的儿子，年仅17岁的Alex Godin只花了7天便学会了编程，并在暑假进入TechStars创业加速器，创立自己的公司。他表示，申请大学和刚进入大学的阶段最为重要。但大学课程并非那么重要，因此是否从大学毕业并不是关键。

05 Google曾为Android寻找Java替代技术

作者 / 水星

7月21日，甲骨文诉Google的Android侵权诉讼案举行听证会，后来公开的听证会记录披露出，Larry Page和Sergey Brin曾要求“调查哪种技术可以成为Android中Java的替代方案”。在内部邮件中，Google员工写道：“寻找了许多方案都不行，因此得出结论：我们要商讨Java授权。”

06 李彦宏百万美金奖励基层员工

作者 / 佚名

8月10日，百度最高奖颁出，金额高达百万美元。此奖项由李彦宏在2010年7月提出，主要针对公司总监级别以下的基层员工，奖励对象为10人以下的小团队，今年获奖的是“智能优惠管理系统”团队，包括基层的技术和产品人员等共计10人。

07 创新工场成长烦恼

作者 / 丁家乐、余涛

2009年9月4日，李开复发布了一条微博：“再见，谷歌”。3天后，创新工场一夜成名。这座工场孵化出了点心、豌豆荚、应用汇、友盟、魔图精灵等创业项目，并将知乎和点点收入麾下，但现在还远未到谈论商业回报的时候。

08 永不被解雇的秘密法则

作者 / NICK O'NEILL

作者指出永不被解雇的秘密法则，即在线写专业领域文章，并分享了四大原因：一、你将成为一个名人；二、它可以使你更加出名；三、你会变成领域专家。四、你构建了一个人际关系网：以前有人曾说你的关系网代表着你的身价。

09 新浪VS.腾讯

作者 / Rip Empson

在中国，新浪和腾讯正在抢占微博市场。双方的获胜优势是什么？哪方将胜出？获胜方是否能最终成为“中国版Twitter”？TechCrunch发文对上述问题作了回答：新浪很可能是未来微博的领军者；但腾讯可能会在更大的平台上胜出：中国社交市场。

10 Google收购摩托罗拉移动

作者 / 付江

8月15日，将深刻影响未来移动互联网战局的收购案得到证实。Google宣布，已与摩托罗拉移动签署最终协议，将以每股40美元的现金收购后者，总价约125亿美元，与摩托罗拉移动此前一周收盘价相比溢价63%。该交易已经得到两家公司董事会的批准。

2011十大职业技能排行榜

职业技能需求分析网站Indeed.com对北美的招聘广告做了分析，结果发现在这些招聘广告中，被提及频率最高的技能是“HTML5”，其次是“移动应用”，排在第三位的是“Android”。

“云计算”和“虚拟化”也排在前10名。可供对比的是，虚拟技能的需求在过去5年中逐渐升温，而HTML5技能的需求则迅速蹿红，仅经过18个月的发展就从无人问津变得广受追捧。

01 HTML5

Web2.0技术让所有人都享受到了技术发展和体验进步的乐趣。作为下一代互联网标准，HTML5自然也是备受期待和瞩目。

02 Mobile App

iPhone的热销及其推动的Mobile App产业正改变整个移动互联网，甚至美国方言协会评出的2010年代表词也正是App。

03 Android

该平台号称是首个为移动终端打造的真正开放和完整的移动软件。受到大量第三方厂商、山寨厂商的欢迎，生态系统繁荣。

04 Twitter

自诩要成为“地球的心跳”，获得大量效仿者的追随，在处理大规模数据方面有相当技术储备，在众多社会运动中发挥作用。

05 jQuery

由美国人John Resig创建，吸引了来自世界各地的众多JavaScript高手加入其团队，其宗旨是写更少的代码，做更多的事情。

06 Facebook

如果它是一个国家，则人口总数排名世界第三位，带领社交网络风靡全球，与Zynga的互生关系，成为互联世界互利共赢的典范。

07 Social Media

不再是Twitter、Facebook，虽然拥有强大的聚合能力，但SNS比起自觉担负起这样使命的应用还是有天生的不足，比如：Flipboard。

08 iPhone

完美的硬件、完美的软件、完美的平台，有人说软硬通吃的Apple只是一个特例，但它的确开创了移动设备软件尖端功能的新纪元。

09 Cloud Computing

通过网络以按需、易扩展的方式获得资源，“云”中的资源是可以无限扩展的，可以随时获取，按需使用，随时扩展，按使用付费。

10 Virtualization

虚拟化使得用户获取数据等资源的时候，不受现有资源的架设方式、地域或物理状态所限制。在异构的云计算领域，获得了空前的发展。

Sencha发布移动HTML5图表库

Sencha发布了Sencha Touch Charts：一套使用HTML5构建并针对移动设备优化过的富客户端、交互式的图表组件的Beta版。作为Sencha Touch的一部分，开发者可以使用该库构建针对Apple iOS、Android以及BlackBerry设备的交互式雷达、柱状、直线、堆叠以及饼状图。

Reds：一个Redis加node.JS的全文搜索引擎

Reds是由LearnBoost公司的TJ Holowaychuk开发的一个基于Redis的node.JS全文搜索引擎，其代码加上注释也只有300行。不得不说又是一个Redis的最佳实践，它的主要原理是通过Redis的sets数据结构，存储分词后的词语碎片。

微软推出开源平台.NET Gadgeteer

微软推出了一个开源软件和开源硬件平台.NET Gadgeteer，但兼容.NET Gadgeteer的硬件价格不菲。.NET Gadgeteer是一套用于创造不同用途的小型电子设备的开源工具集，使用.NET Micro Framework和Visual Studio/Visual C# Express，结合硬件模块和.NET软件，让用户能在不十分了解硬件知识的情况下，在数小时内创造出智能电子设备，制造出快速原型设备，帮助教师设计新颖的交互教育仪器，帮助业余爱好者创造出想象中的事物。首个兼容.NET Gadgeteer的硬件模块是GHI Electronics的Fez Spider kit，完整工具包售价249.95美元。

Google开源LevelDB

Google宣布在BSD许可证下开源其键值存储引擎LevelDB。 LevelDB C++库可用于多种不同环境，如被浏览器用于存储最近访问的网页缓存，或者被操作系统使用去储存安装的软件包和依赖包清单，或被应用程序用于存储用户设置。Google称，即将发布的新版Chrome浏览器，就包含了基于LevelDB的IndexedDB HTML5 API实现。Google Big Table曾使用了LevelDB的前身，而Riak分布式数据库已经加入了LevelDB支持。

英特尔宣布开源Cilk Plus

Cilk Plus是C和C++编程语言的扩展，是为多线程并行计算而设计的，它允许C和C++程序员高效利用多核处理器的并行处理能力。Cilk多线程编程技术最早由MIT开发，是一个基于GCC编译器的开源项目。后来开发者创建了一个创业公司，推出改进的私有版本，整合到Windows下的多种编译器中。之后它被英特尔公司收购，整合进英特尔的编译器中。现在，它再次成为一个开源项目，成为GCC 4.7下的一个分支。

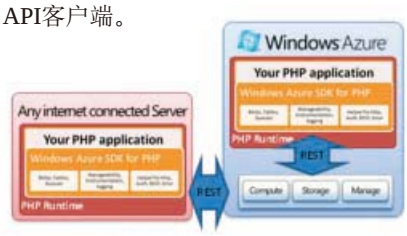
SilveOS：基于Silverlight的Web操作系统

SilveOS的界面照顾Windows用户的习惯，内置文件浏览器、网络浏览器、多媒体播放器、记事本，以及一些网络服务如Twitter、YouTube、Virtual Earth，同时还有一些游戏以及聊天画图和计算等小工具。

用户可以用Guest帐户查看和试用SilveOS，也可以免费注册一个账户。

微软发布Windows Azure SDK 4

在微软发布Windows Azure SDK for PHP 3.0.0的短短几个月后，也相继正式发布了Windows Azure SDK for PHP v4完整版。Windows Azure SDK 4包含了一些重要的特性和改进，它集成了一个用于访问Windows Azure存储的PHP库、logging组件、会话共享组件、面向Azure以及SQL Azure管理的API客户端。



Adobe推出Muse网站创作工具

Adobe发布了代号为Muse（Code Name）的Web设计软件，该工具是面向图形设计师的站点创作工具。继本月初发布Adobe Edge动画工具，这是Adobe发布的又一款全新的网页设计工具。Adobe官方人士称，Muse能让平面设计师设计和发布专业品质的HTML网站，而无需手工编写代码或仅局限于模板内工作，Muse号称能让Web设计师从代码中彻底解脱出来。Muse支持最新的Web标准，包括HTML5和CSS3。另外，Muse与Adobe InDesign可以很好地相结合，创作出具有交互式内容和各种流行元素的Web站点。

Google宣布Google CDN

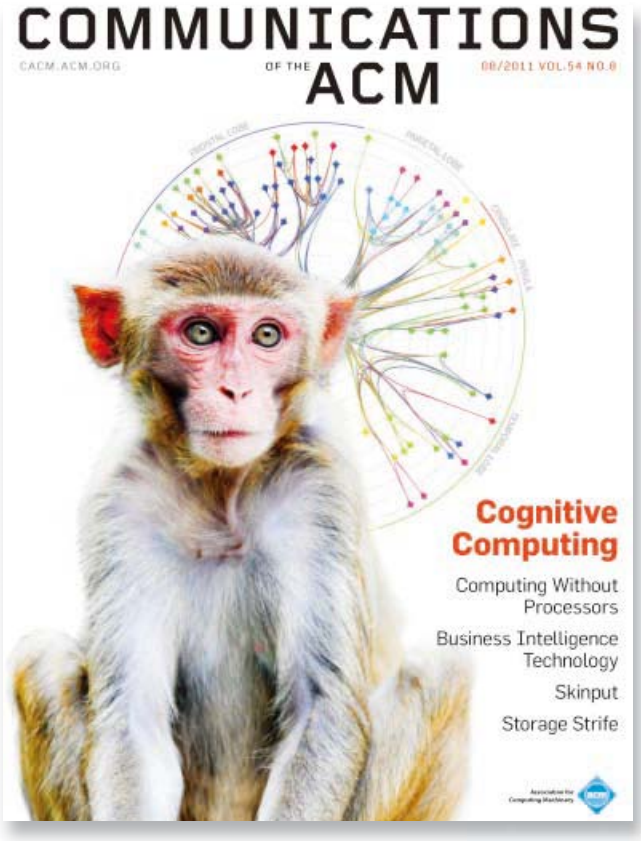
Google宣布了最新的帮助加快互联网速度的工具Page Speed Service，加快静态网页的载入速度，不支持动态网页。 在开发者注册该服务之后，可将网站的DNS入口记录指向Google，然后Page Speed Service从服务器上抓取内容，采用最佳的Web性能方案重写网页，通过Google在全球部署的服务器将内容展示给终端用户，加快网页载入速度。网站开发者将无需再担心CSS串联、压缩图像、缓存、压缩资源等问题。Google称它的测试显示能加快网页载入速度25%~60%。这项服务在Beta测试阶段免费，收费细节将在以后公布。

Adobe Edge发布

Adobe发布了Edge Preview 1，允许Web开发者使用HTML5、JavaScript和CSS3等Web标准，为网站开发动画和交互内容。Adobe Edge使用了开源JavaScript库jQuery。jQuery是Edge的重要组件，Edge Preview 1使用jQuery 1.4.2，后续版本将使用较新版的jQuery，Adobe发布的下一个版本预计将使用jQuery 1.6.2。

starflow工作流引擎 0.7.0发布

starflow是一款富有中国特色的工作流引擎，刚刚发布了0.7.0版本。作为一款轻量级的工作流引擎，starflow融入了电信行业业务流程的特殊要求，提供了灵活的工作任务分派策略、业务流程版本管理策略、丰富的流程模式、灵活的组织模型和子流程等特性。webframe是starflow工作流的一个Web应用工程，包含系统管理和工作流管理。



Communication of ACM

2011.08

反思鲁棒性原则

1981年，乔恩•伯斯塔尔提出了鲁棒性原则，也就是伯斯塔尔法则，作为当时新出现的TCP的基本执行原则。鲁棒性原则提倡在产生并发送协议时，需要尽可能保守地遵循规范；接收协议时，则需要尽可能地开放。其意在最大化网络服务之间的互操作性，特别是在规范不明确和不完整的情况下。

虽然鲁棒性原则是为了TCP而特别提出的，但它的主张很快被接受，成为了在执行一般网络协议时的普遍原则。它被应用在API，甚至是编程语言的设计中。这个主张简单直观，容易理解。多年以来，其应用也颇显成效，但它也并非绝对真理。近年来，随着外部环境的变化，鲁棒性原则对互操作性和安全性造成了冲击。

网络协议执行中的互操作是一个难题。规范的存在就是为了让互操作能够顺利进行，规范必须精准且明确。但为了使其具有可扩展性，规范中常常存在一些模糊性。鲁棒性原则让具有模糊性的规范也能很好地被执行，但同时，也让很多不正确的执行方法孳生。

在如今复杂的网络环境下，“在接收时保持开放”很容易造成安全隐患。事实上，保证软件可靠性和安全性的一个重要原则就是检查每一次输入，这意味着“在接收时要保持严格”。

以上论述以TCP等网络协议为例，但实际上，开放性过剩的情况，普遍存在于MIME、网络服务器、小型计算机、Sendmail程序和网络浏览器中。

但这并不意味着鲁棒性原则本身就是错的，也不是说我们应该完全严格地执行规范。“在接收时保持开放”，这一点让我们能够扩展协议。无论再成功的协议，总有需要扩展的一天，可能是最初的设计没有考虑某些问题，或者是因为外部环境的改变所致。世界永远是变化的，规范——以及规范的执行——要充分考虑变化和危险。但我们在应用鲁棒性原则时，必须适度。

认知计算

意识是什么？没有哪个科学家或者哲学家能给出一个公认的确切答案。一般来说，我们把意识看做是感知、理解、行动、情绪以及认知等一系列过程的集合。意识可以将由听觉、视觉、触觉、嗅觉和味觉而来的模糊的信息结合起来，建立它们之间的时空关联，形成抽象的概念，做出决策并引导复杂的协调动作。

认知计算的目的是寻找意识中一致而统一的普遍机理。许多不同领域的研究者采取截然不同的方法来模仿意识的过程：人工智能、神经系统科学、认知神经科学。这些领域的繁荣发展，让我们从不同的抽象水平上认识意识。

在这种背景下，本文提出了一种新的研究方法，在大量的神经科学数据基础上进行大规模计算仿真。利用神经生理学和神经解剖学研究的成果，在小哺乳动物的大脑水平上，进行接近实时的仿真，以求解释和实现大脑的核心算法，揭示意识产生的过程。

意识的产生过程有很多可能的方式，构成了一个庞大的计算空间。大脑是认知计算的一个解决方案样本，因此神经科学对大脑的认识可以提供一些限定，有效地缩小搜索空间。比如说，神经解剖学对神经网络结构的认识，以及神经生理学对大脑信息处理和认知功能的基础——神经细胞功能的认识。

我们的工作是从这些限定中选取合适的子集，综合起来构成一个计算平台，一个哺乳动物大脑规模的模拟器。研究人员使用了蓝色基因超级计算系统，并开发了C2皮质模拟器，采用分布式内存多处理器架构，以满足大脑模拟对规模、速度和细节水平的要求。

由于冯·诺依曼结构和大脑结构的差异，模拟需要的资源远远大于大脑本身。这项研究目前还没有取得突破性的成果，但随着哲学、神经解剖学、神经生理学、计算神经科学、超级计算和计算架构等学科的进展，实现对人类大脑规模的皮质模拟指日可待。

商务智能技术

商务智能（BI）软件是企业中决策支持技术的集合，目的是让知识型工作者，比如行政主管、经理以及分析员，更快更好地做出决定。过去20

年中，随着大量数据获取和存储成本的下降，BI产品和服务的数量，以及这些技术在工业中的应用，都有了爆炸性的增长。不管是在产业应用还是在研究领域，商务智能的格局都经历着巨大的变化。

数据的获取越来越容易，数据定期地被收集，数据量非常大。企业大量地使用这些数据资产，部署和尝试复杂的数据分析技术，来推动企业决策或供个性化的服务。这些数据来自许多不同来源，质量也参差不齐，需要整合、筛选以及标准化。另外，高效和可扩展的数据装载和更新能力也是企业BI所必须的。这些进行数据准备的后端技术，统一被称为ETL（数据抽取、转换和装载）工具。

这些数据将会保存在数据仓库里。最普及的数据仓库服务器是关系数据库管理系统（RDBMS）。商务智能要求在大容量的数据中执行复杂的SQL查询，过去20年中，针对这一点，开发了很多数据结构优化和查询处理技术。大型的数据仓库通常部署并行的RDBMS引擎，这样SQL查询可以以较低的延迟在大量的数据中执行SQL查询。由于传统的RDBMS无法满足处理逐渐增多的数字化数据的要求，因此，基于MapReduce范例的引擎——最初用于分析网络文档和网络搜索查询日志——现在也被用于企业分析，经过扩展后，支持传统的企业数据仓库情境中复杂的类SQL查询。

中间层服务器是数据仓库的补足，为不同的BI情境提供特殊的功能，比如说OLAP、内存BI引擎、报表服务器等。另外，文本数据也越来越多地被作为有效的商务智能资源，企业搜索引擎开始更好地支持文本，专门的文本分析引擎也得到开发。

一些前端应用也帮助用户更好地完成商务智能，比如电子表格、企业搜索门户、绩效管理应用、数据挖掘模型查看器等。快速而专门的数据可视化也被用来动态化探索数据中的模式，解释它们之间的关联。

另外，还有其他一些商务智能技术得到了应用，比如网站分析，可以帮助了解公司网站的访问者如何与页面互动；像CRM这样的垂直打包应用也得到广泛应用，这类应用通常支持内置分析。随着云数据服务逐渐站稳脚跟，会为BI后端架构带来更多变化。而另一个在萌芽中但很重要的趋势是移动BI，知识型工作者对移动设备上的互动BI需求会越来越大。

Wired

2011.08

创新并非凭空而来

技术迷们喜欢谈论重大创新——那些突然出现然后迅速风靡的产品和技术，像是iPhone和Twitter。有很多评论家、投资者和网站兴奋地预测着下一个将会出现的重大创新，而且这个创新一定又是哪个天才灵机一动想到的出其不意的点子。

但Bill Buxton，一位计算机图形的先驱，微软现在的首席研究员却不这么看待。关于创新，他有一个“长鼻”理论：任何即将造成巨大影响的创新，通常已经诞生并存在了将近10年时间。

以苹果iPhone上引入的“捏小拉大”（pinch-and-zoom）手势为例：这看起来像是一项横空出世的全新技术，但实际上，1983年，计算机设计师Myron Krueger 就首先在他实验性的VideoPlace系统中运用了该手势。其他工程师和公司受其启发也开始陆续尝试。到iPhone推出这项功能时，手势控制实际上已经是成熟的概念了。

另一个最近的例子是微软的Kinect。没错，仅凭舞动身体来控制软件的想法听起来很新鲜，但工程师们对动作传感的完善已经有很长一段时间了。我们用身体来控制软件已经很多年了，只不过是在其他的

领域。

这就是为什么能带来上十亿商机的重大创新实际上也非常显而易见。它们让人感觉既新鲜又熟悉。就是这两点的结合让它们能够迅速起飞，攻城掠地。在设计iPhone时，苹果的设计师们从PDA市场的成功和失败中汲取了经验教训。他们增加了触控手势，成功地化平凡为新奇。

如果你想预测下一个重大创新，你要做的就是“挖掘”，去寻找在某个领域里成功的概念，然后把它们应用在另一个领域。Buxton预测平板电脑、笔式界面（pen-based interface）和电子墨水在今后10年会大展身手，因为在缓慢地发展了20年之后，它们已经准备好了。

社交机器人

在这篇文章中，Steven Levy探讨了机器模仿人类行为并代替我们发布社交网络信息的可能性。在微博上，我们的兴趣总是在某几个限定的点上，经常会重复地访问几个类似的话题。

这种重复性让作者想到了：是否可以创建一个程序模仿人们在社交网络上的行为，替我们发布微博。这种“自动驾驶仪”还可以分析人们在Facebook、Twitter、Foursquare等社交网络上的数据，了解这个人的兴趣、喜欢推荐什么样的东西、交流时惯用的腔调，从而不断地完善其能力，成为我们在网络上的“代理人”。

最理想的情况是，没有人能发现是自动程序在发布信息。即使我们哪天遭遇不测，程序还会继续工作，我们一部分的意志还继续活跃在网络上。

1950年，艾伦•图灵在一篇论文中预测，有一天，计算机会智能到足以冒充真人，并且提出了鉴别的方法。从那以后，图灵测试就成了人工智能的金科玉律，人们每年都会举办图灵测试的竞赛。但Brian Christian在他的新书《最像人的人》（*The Most Human Human*）中说道：图灵测试夸张的机制——让测试者和被测试者开展5分钟的对话——没有考虑到人类在自然情况下的反应方式。

如果把人工智能的测试方式改成：社交网络自动程序是否能够以假乱真地模仿人类在社交网络上的行为，将是件有趣的事情。David Ferrucci以IBM的智能计算机Watson在电视节目Jeopardy的智力竞赛中战胜人类冠军为例说明，这是件非常可能的事情，而且自动程序会表现得非常好。

虽然自动程序也会有表现失常的时候，但随着软件的不断改进，这种失误会越来越少，自动程序的表现甚至会超越人类。

社会学家Sherry Turkle观察到，我们在社交网络中表现的人格不是人们真正的自我，而是他们心目中理想化的自己，这就像是在戏中扮演一个角色。但是人们很难区别真实的生活细节和想要创造的角色，所以要“扮演”理想的自己有点困难。

但是计算机没有自我意识，因此社交机器人反而能比人类更善于扮演这个角色。

TaskRabbit 把勤杂工作变成游戏

爱迪生有一句名言：天才，是1%的灵感加上99%的汗水。但Leah Busque的革新型创业公司TaskRabbit则是因为懒惰而建立的。你可以把这家公司看做是交易劳动力的eBay，“发令者”发布任务，以及他们肯为这项任务支付的最大金额。“跑腿者”则声明自己能接受的最低报价，竞拍这项任务。

这家网站成立于2008年9月，到今天，该网站已经在旧金山、波士顿、洛杉矶和奥兰治县拥有1500名跑腿者，每月完成3000项各式各样的任务——从组装宜家家具到组织啤酒聚会。

对于网站用户（发令者）来说，TaskRabbit的跑腿者就像是随叫随到的帮手。因为有大部分任务需要在发令者家中完成，跑腿者必须经过三步审查：提交申请表，通过电话或视频面试，最后TaskRabbit还会雇佣数据库公司Acxiom调查每个人的联邦犯罪背景。

对那些处于长期失业状态的“跑腿者”来说，该网站提供了一种不影响自主权的稳定收入来源。TaskRabbit允许工作者直接被客户雇佣，并建立他们自己的声誉。它帮助工作者成为自己的工作经纪人，让他们有机会开创自己的商业模式。

为了保持网站的活力，TaskRabbit引入了一些游戏机制。在跑腿者排行榜上，列出跑腿者的级别和平均用户评分。还会用一个和游戏里一样的进度条，显示晋升到下一级别所需的点数。准确和快速的竞拍，以及邀请好友加入网站，都能获得相应的点数。达到一定级别后，网站还会给予一些现实中的奖励。

每个月，TaskRabbit会主办一次聚会，让这些游戏者/工作者可以互相交际。

目前TaskRabbit和它的竞争者（AirRun以及Zaarly）都还处于很小的规模。它们是否能扩大规模，成为一大批劳动力的谋生手段，还是个问题。

但是TaskRabbit把工作和游戏相结合的做法，确实指出了一种新的雇佣方式，也许能开创一种不同的企业形式。

（感谢译者卞斌支持）

Oracle的努力

Oracle为MySQL做的事，对于社区利弊几何呢？也许要在一个更大的时间跨度上去考量。

8月关于Oracle的消息非常多，除了照例一系列的收购外，关注度最高的恐怕就是Java 7的准点发布，不过包括一系列新增特性的MySQL 5.6官方早期版本的发布同样也很抢镜。回想当年，欧盟对Oracle收购Sun延期评估最大的口实就是担心Oracle可能会趁机扼杀MySQL，进而垄断市场，但Oracle似乎另辟蹊径，从新的技术模式中找到了改进和进一步支持开源MySQL产品的动力，“NoSQL+云计算”。

当作为嫁妆随Sun一起“嫁入”Oracle时，MySQL已经具有了比较完整的备份/恢复功能和一定的复制能力，也就是说它跨进了企业应用对于高可用和容灾要求的门槛，尽管相比Oracle数据库还有一定差距，但凭借性价比的优势，MySQL也可以在很多企业场景中独当一面。

完成收购后，Oracle坚持采用InnoDB作为MySQL默认的存储引擎，此次升级还专门在全文检索查询的效率上了下不少功夫，尤其对“写密集”（write-intensive）的应用，力图提升高并发情况下的用户响应。此外，Oracle还将MySQL 5.6的Redo Log文件容量上限从2GB直接提高到4TB，确保MySQL能够支持更多的长交易/长事务（Long Running Tx）。

此外，作为MySQL复制技术底层API的Binlog，同样在新版本中得到加强。通过Binlog，开发团队不仅可以实现同版本MySQL数据库、不同版本MySQL数据库间的数据同步，还可以与各种非MySQL数据库同步。由于Binlog位于MySQL程序栈中较为底层的位置，因此尽管API比较原始，但效率更好。Web 2.0应用可以通过Binlog API实现与NoSQL数据库的联动，合理分配结构化在线数据和NoSQL近线、远线甚至离线数据的使用，最大程度节省运行成本。5.6版本还在原有Binlog API的基础上增加了group commit功能，其最大的作用就是可以充分利用系统并行处理能力，这也为分布式存储系统提供便利。

除了技术上的改进外，商业上为了保持MySQL的开源特征，上述功能也是由Oracle的MySQL工程团队完成的，所有的功能都遵循GPL协议开源，因此开源社区可以测试和深入地跟踪调试Oracle MySQL 5.6，便于借助社区的力量检验MySQL这些“登堂入室”的企业版功能。

粗看这些特性，可能只是Oracle对MySQL在扩展、性能和灵活性方面做的零散改进，但如果结合前期Oracle面向MySQL集群提供的memcached API来看，这可是Oracle面向NoSQL和时下火热的云计算走的一步好棋。区别于Oracle数据库，MySQL本身是一款开源产品，历史的“坛坛罐罐”不多，并且可以和Linux平台以及Oracle自己的虚拟机产品（Oracle VM）很好地集成，因此对于传统企业或大型Web 2.0企业而言，与其自己花心思维护大型的私有云，也许不如采用“Oracle MySQL数据库 + memcached API + Binlog API”的方案更节省部署时间，并且由于有MySQL社区和专门开发团队的不断完善，用户相当于掏着开源软件的钱、用着商用产品的功能。尽管很多其他NoSQL数据库在某些方面有一定优势，但这些产品多多少少有些“独”，也就是说一旦用某款一段时间，即使发现软件本身有“硬伤”，想也很难迁移，而MySQL不同，由于其前期用户基数大、普及率高，被Oracle收购后，与其他数据库产品多个层次互联互通方面也做了一定改进，因此比较适合作为云计算环境中的中心数据库。由它管理结构型数据并通过底层API与其他关系数据库、NoSQL数据库同步。甚至整体上看 MySQL自身已经在向“关系数据库 + NoSQL数据库”的2in1产品过渡。

Oracle为MySQL付出的努力从中短期看对社区和用户是有益的，因为用户可以用很低的软件购买成本（不计运维、培训、咨询费用的话）获得这些产品和企业级特性；但从长期看，对于社区的效果就比较模糊了，因为Oracle等于在用职业队和业余队PK，预期将进一步加大Oracle的市场份额，从“一家独大”到“独一家大”。

王翔
软件架构师，主要研究方向为XML、.NET、领域设计和PKI应用。工作之余喜爱旅游、写作和烹饪。

Google收购摩托罗拉

Google宣布以125亿美元收购摩托罗拉移动，也许意味着一个时代的终结和一个更宏大时代的开篇。

在移动互联网漫长的历史上，能够称之为转折点的事件并不在少数，但标志着移动操作系统布局阶段完成、进入混战时代的事件却只有一个——Google宣布以125亿美元收购摩托罗拉移动。

摩托罗拉创立于1928年，从第一部寻呼机、第一部手机，到野心勃勃的铱星计划，可以说摩托罗拉的历史就是手机的编年史。Google创立于1998年，以搜索引擎闻名于世，通过Android操作系统进入移动设备操作系统市场，通过免费开源的方式提供操作系统，改变了移动操作系统被少数强势公司统治的格局。这一次的收购，能够与AOL收购时代华纳相提并论，但结果是否会比那次好一些呢？

如果把移动互联网巨头们之间的角力看作一盘很大的围棋，那么Google收购摩托罗拉之前的阶段属于布局：金边银角草肚皮。Apple占了高端智能手机市场，Android在中低端智能机市场看似无可匹敌，微软在二者的夹缝间辛苦地寻找着自己的出路，而RIM、webOS、Symbian还在各自的细分市场上深耕细作。虽然时而爆发一些局部冲突，但总的来说，移动操作系统的格局还是稳定的。

打破这种格局的是Android，在经历了三年多“野蛮生长”之后，Android已经渗入了移动互联网的各个领域，从智能手机，到平板电脑，然后是互联网电视、车载电脑。这种野蛮生长的原因其实也很简单——生逢其时。Android生于风云际会之世，旧的势力如Symbian、Windows Mobile还没有从授权收费的迷梦中醒过来，而新势力如webOS、MeeGo，还没有真正成熟。因此，本身不做硬件、免费、开源的Android才能够横扫所有竞争对手，成为手机厂商的不二选择。

Android用七百人做了微软三千人做的事情，很长时间以来，这是Android社区笑话微软的经典段子。Android的确是工程学上的奇迹，利用了很多开源社区的技术，在最短时间内搭建出完整的技术体系，包括：操作系统、开发工具、SDK、驱动程序架构。但付出的代价是丧失整个产业链的控制权。当大量的芯片厂商、硬件厂商为Android贡献了非常多的代码时，Google对于整个产业链的控制就相应减弱了。说白了，加什么，或者减什么，不是Google一家说了算的。

松散联盟的特点是，好的时候，众人拾柴；坏的时候，墙倒众人推；人人都想从中获得利益，却不想为之贡献力量。Symbian当年也只有诺基亚一家出死力。

快速发展中，Android的弊端也逐渐暴露出来，首先是兼容性问题，由于开放给手机厂商的权限过大，每家厂商都以定制自己的系统为目标；然后是专利问题，由于Google缺少在移动领域的专利积累，Android厂商成为了其他专利大鳄眼中的美食。这一次，促使Google收购摩托罗拉的真正原因，正是Apple和微软发起的专利超限战。

专利，一方面是后发公司眼中的紧箍咒，另一方面也是促使先发公司投入基础研发的催化剂。在专利保护不佳的地区，往往会出现市场极度繁荣，而基础研发却乏人问津的奇怪现象，长期来看，专利对于促进基础研发有着非常正面的意义。

如果真如Google所说，收购摩托罗拉是为了解决Android的专利授权问题，的确，Google可以通过专利交叉授权的方式，免除HTC、三星、索爱等一线厂商的专利陷阱。但对于与Google不存在授权协议的二线厂商、山寨厂商，专利反而成了它们头上的达摩克利斯之剑，再也不会有Google这样的带头大哥代为出头了。更为尴尬的是那些使用了Android代码却号称“自主研发”的类Android系统们，本来和Android极力撇清关系的它们，如今要么回来抱Google的大腿，要么独自面对专利大棒的绞杀。

Google收购摩托罗拉后，其他各方的反应如何呢？三星收购了Android ROM定制团队，并且宣布加强bada的研发力度。移动操作系统的迁移，从研发到稳定，大概需要两年的时间，三星、HTC都不能抛弃在Android上花费的大量投入。但耐人寻味的是，三星和HTC始终都没有放弃Windows Phone产品线，不把鸡蛋放在一个篮子里，大概是这些一线厂商行走江湖的无上秘诀。

Windows Phone一夜之间成为唯一不由本家生产硬件的移动操作系统。如果说，保持操作系统独立性是取外

势的话，那么收购手机厂商，就是占实地。如果在Google收购摩托罗拉之前，微软收购诺基亚，算是一步好棋，软硬合一，诺基亚的Windows Phone 7取下20%左右市场份额应该不算难事。但Google收购摩托罗拉之后，如果微软再采取类似行动，那就是逼HTC、三星研发自己的操作系统了。

故，孙子曰，兵无常势，水无常形；能因敌变化而取胜者，谓之神。

马宁
微软最有价值专家，Windows Phone开发者。

函数式编程语言 F#

作为微软支持的第一个函数式语言，F#在项目中被越来越多的开发者选用，8月的TIOBE排行榜，F#挺进前二十。

源于微软研究院的F#语言因其优良的设计和强大的并行编程能力，正得到越来越多.NET开发者的选用。在8月的TIOBE语言流行度排行榜中，F#语言首次进入了前二十位。F#是微软.NET框架环境下的静态类型化函数式编程语言，支持以面向对象或泛型编程等多种风格来撰写程序代码。

F#基于函数式编程语言Objective Caml（OCaml）设计，具有OCaml常用的核心语言功能，以及函数式编程语言的其他特性。OCaml是在Perl之外，又一门以骆驼为吉祥物的编程语言。Ocaml被选中的部分原因是它提供了函数式、命令式和面向对象的混合编程风格支持。在OCaml语言中，函数式能够像变量一样方便地在程序中传递，并且OCaml是能够自动侦测程序范型的函数式编程语言。

F#语言最初由微软研究院的首席研究员Don Syme设计，设计者Don Syme于1999年从剑桥大学计算机实验室获得博士学位，他还参与了C#泛型和.NET CLR的设计工作。在语法设计上F#具备优雅的结构，同时F#被设计为类型安全且具备良好性能的编译语言。在微软决定将F#进一步推广应用之后，F#被转移到微软专门的开发部门维护和更新，并在.NET Framework和Visual Studio 2010开发环境中为F#提供了全面支持。此外，开发者还能够在Visual Studio 2008中使用F#，或借助Mono在Linux系统上使用F#。

作为微软支持的第一个函数式语言，F#在项目中被越来越多的开发者选用，这与F#对程序并发和异步编程的良好支持密不可分。微软为F#添加了不少简化开发者处理程序并行和异步编程的便捷特性，以辅助开发者轻松完成多核并发和Web分布式系统的应用开发。在常见的开发模式下，F#用于撰写安全并发和异步处理相关的程序组件，用户界面则使用Visual Basic或C#来编写，而最终产品由F#实现组件和其他.NET开发语言组织而成。

不仅如此，F#目前还广泛用于Xbox 360游戏平台上的代码开发中，为开发者提供了在XNA Game Studio环境下的F#编程支持。微软研究院曾使用F#、TrueSkill以及XNA开发了一款名为“The Path of Go”、具备故事情节的3D围棋游戏，以展示F#在游戏开发和人工智能领域方面的并发编程能力。

感兴趣的开发者可以阅读Don Syme及其他几位F#语言设计者共同撰写的《Expert F# 2.0》一书，深入学习并使用F#语言。也可以阅读游戏开发者Giuseppe Maggiore撰写的*FRIENDLY F# with game development and XNA*一书，学习F#语言在游戏开发中的应用。

高昂
中国标准化研究院助理研究员，从事信息技术标准化研究工作。关注开源社区，也是OSGeo中国和InfoQ中文站成员。

B2C不挣钱，O2O新概念

B2C也好，O2O也好，终究也许都要回到盈利的道路上来，虽然前途并不平坦。

在刚刚结束的2011派代电子商务年会上，看到某电商CEO这样一番言论：电商企业未必一定要挣钱，他总结说，电商企业如果是自己的钱，就不要想去亏；如果是投资者的钱，就拼命去亏。这句话也正是京东商城目前的真实写照：反正是VC的钱，拼命亏去吧。B2C是否挣钱这个话题，从当当网十年没有盈利，持续到现在，已经不觉新鲜，京东商城刘强东也一直以京东不盈利作为公司的考核目标，而被人斥为不讲商业道德。好乐买CEO李树斌在派代电子商务年会上也表示，电子商务是可以收放的，如果大家每年都保持300%的增长，亏损会一直持续下去；如果保持130%的增长，很多公司就盈利。大家需要考虑最后到底需要300%的增长不盈利，还是130%的增长盈利。

B2C网站到底是什么东西？有人说，它首先是一个贸易型的公司，如果按照这个思路去理解，那么所有的人都会惊异VC们的人傻、钱多，持续十几年投资一个并不赚钱的行当。不要说最近几年，京东商城已经持续融资了很多亿美元，就是最新的奢侈品网站尚品网和走秀网之类的，也在今年短短半年内，分别融资多轮，达五千万美元乃至几个亿美元。说个题外话，有几个软件公司能有这么大的手笔呢？这使我想到了可怜的上海普元，拿了一点钱，也奋斗了很多年，但要想像B2C网站这样融资，依然是痴人说梦。能做VC的人，无疑是这个世界上最聪明的，如果仅仅把B2C网站看作是个贸易型的公司，仅仅是把货品摆在网上卖，那么就真的无解了。在现实生活中，就看看我们周围的超市、家居旗舰店之类的，如果半年一年不盈利，肯定早就关门了，肯定不会容忍你多年不盈利，更不会说十年以上持续不盈利。个人认为，B2C网站首先是一家互联网企业。互联网在媒体、新闻上的行业化，就是新浪、搜狐和网易，尤其是新浪，一家典型的媒体型公司。互联网在交易上的行业化，就是阿里巴巴、慧聪和中国制造网之类的公司；相应的，互联网在面向消费者购物上的行业化，就是B2C、C2C网站。例如京东、当当和淘宝。

如果不把B2C网站，看作是一个互联网企业，很多行为都是无法理解的。京东、当当每年都在百度投放数千万的广告费，如果仅仅是提高知名度，投中央电视台可能更权威，投放百度都不知道投放到哪里去了。购物流量巨大的淘宝网，分别在流量最大的新浪网和环球网，花巨资包了固定的CPM广告位，所以，B2C网站首先应该定位于一家互联网公司，经过20世纪90年代互联网泡沫的洗礼和教育，那么大家也就都心照不宣了，互联网烧钱多年不盈利也就不难理解。互联网的战争就是入口的战争，互联网的基因也就是烧钱买流量（花VC的钱），然后再卖流量（卖广告）。所以京东、当当、凡客、卓越亚马逊在融巨资弄来流量后，也都在出租自己的电子柜台。

至于O2O，百度百科的解释是O2O即Online To Offline，也即将线下商务的机会与互联网结合在了一起，让互联网成为线下交易的前台。对于O2O，我在微博上看到如下一段精彩的评论：才听说有O2O这个模式概念，今天了解了一下才发现，十年前开始的旅游电子商务不就是O2O吗？ 典型的线上购买，线下享受服务，这个早已被各大旅行社应用娴熟。演唱会、音乐会等票务也是如此，七八年前就有了，不得不佩服这些人的智慧，捣鼓捣鼓就弄一个新模式忽悠VC。事实上，Groupon这个团购行业的鼻祖（虽然才存在不足3年，最早成立于2008年11月），也是备受争议的，有人甚至说他是庞式骗局。上市之路也是不平坦的。

个人认为，B2C也好，O2O也好，终究都是要回到盈利的道路上来，什么时候能够盈利，怎么才能够盈利，不是一下子能够讨论清楚的。否则，争议也不会持续十几年，无论他们能否最终修成正果，但无疑的，贸易互联网化之后，还是给大家的生活带来了革命性的影响。每每走到北京地铁一号线和十号线的交叉点，看到凡客的“凡客体”大屏广告，都会由衷地微笑：我是凡客，会写程序、会写评论、会出书，不管你信不信，反正我信了。

邢波涛
北京新软孚信息技术有限公司技术负责人。关注SaaS管理软件和B2B、B2C电子商务的融合。

工控告急

如果国内安全厂商在平时不能对工控做构建等准备，一旦出现安全事故再进场，必然十分茫然。

8月有一件事让我感到遗憾——在高铁事故调查组中没有见到信息安全专家的身影。虽然狭义上看这并非一起信息安全事故，但我们应该看到，由物理安全和电气安全组成的现有工业系统安全观已经陈旧，面对复杂的国际形势和国内情况，中国高铁的安全以至整个工业体系的安全都需要经得起国土安全角度的考察和推敲。

这种检视不但要基于常态运营，更要基于突发事件、自然灾害、恐怖袭击等。潜在的威胁不仅来自物理层面上，还可能来自信号层面和信息层面，此时就需要信息安全专家的出场了。

在动车事故发生前，高铁系统已经发生了3起电气事故。网上曾出现传言称这几起事故是某些国家释放出蠕虫所致。我对此做了搜索对比，发现这一传言是由此前在《新京报》的一篇报道中的部分文字与一些所谓“蠕虫”的消息拼接而成。从内容上看，漏洞百出，质量极低。但就是这样一条假新闻，却被国内网站大量转载，其中不乏专业的电气技术协会组织。

全球工业系统的安全形势已经是危机四伏。该领域代表性企业西门子公司的产品必然是攻防双方的一个重点战场。在8月6日的Black Hat USA大会上，安全专家Dillon Beresford介绍了在西门子公司工业控制系统中发现更多漏洞的情况，这些漏洞包括复活节彩蛋、可用于发起远程拒绝服务攻击的漏洞，甚至是管理账号和密码硬编码漏洞。

应当指出，在2010年的震网（Stuxnet）蠕虫事件中，将用户名和口令硬编码到应用程序中的问题已经出现在了西门子公司的WinCC产品中，并成为震网蠕虫攻击的重要环节。这种不符合基本开发规范的编码实现，也意味着攻击者一旦发现并利用，就可以通吃通杀，而防守方却无法彻底解决问题。

震网蠕虫事件时，我们曾指出数据、配置、代码三分开也是工控系统开发的基本原则。但可以看到，西门子公司并未对相关问题做出调整。这似乎表明西门子公司仍然用产品私密性来保障其体系的安全。他们是否还以为，将更多权限开放给用户并不是应对安全问题的有效方法，而只会给自己带来更多麻烦？如果西门子公司没有其他层面的蓄意，我们只能认为其安全观是落后的。

在8月召开的中国计算机网络安全年会上，组织者出于对工控问题的重视，无论是在高峰论坛还是在专题技术报告均安排了西门子公司发言。但西门子公司却将此视为危机公关的机会。

概括其主旨观点，就是“微软的漏洞太多，震网蠕虫作者手段太高超，我们已经做出了响应，所以我们没有任何责任”。至于西门子公司工控系统的漏洞问题，以及对全局问题的总结和反思，则丝毫未谈。这种推卸责任的态度让很多在场的安全界同行愤愤不平。

从全球范围看，目前对工控安全的研究已经从最初对终端系统和应用软件的漏洞挖掘开始向软硬件结合和工控体系安全的方向扩展。今年3月，Ruben Santamarta在RootedCon作了题为*SCADA Trojans: Attacking the Grid*的技术报告，他对电力体系的理解之深刻，让我一位多年从事硬件研发的同事赞叹不已。工业控制系统的基础设施依然是复杂而昂贵的，如果国内安全厂商在平时不能对这些场景做构建等准备，一旦出现安全事故再进场，必然十分茫然。

在这一领域，我们需要感谢US-CERT的工业控制系统安全小组，他们分析整理了大量相关漏洞信息，其简报也是该领域最为系统的聚合信息之一。美国国家标准和技术研究所发布的《工业控制系统安全指南》和《工业控制系统反病毒软件指南》等也是值得研究参考的文献。美国能源局、特情局、国家实验室等也纷纷开通专题网站、发布研究报告，对此问题的重视程度可见一斑。

在国内，CNVD（国家信息安全漏洞共享平台）对工控系统漏洞的及时通报、电力科学研究院的相关标准研究、国内安全企业对震网蠕虫的跟进分析等，也让我国的工控安全事业有了一个自己的起点。

肖新光
网名江海客，安天实验室首席技术架构师，研究方向为反病毒和计算机犯罪取证等。微博：weibo.com/seak。

重返，更为超越

盐湖城敏捷十周年大会纪要

记者 / 熊妍妍

为纪念，而重聚

2001年2月11日~13日，美国犹他州瓦萨琪山脉的雪鸟滑雪山庄内，17人结伴同游，人群囊括了极限编程、Scrum、动态系统开发方法、适应性软件开发、水晶方法论、特性驱动开发、实效编程的代表，还包括在为文档驱动重型软件开发过程寻找替代方案的同道中人。滑雪放松之余他们也彼此交谈，试图达成共识。敏捷软件开发宣言正是在此问世。

图1　为纪念敏捷宣言签署十周年而启动的敏捷宣言翻译项目，由Henrik Kniberg和Ward Cunningham发起

图2　敏捷宣言中文简体字版（由徐毅组织翻译），和其他30个语言版本的敏捷宣言，被一一贴到了大会会场的走廊中

十年后，在敏捷宣言签署十周年之际，敏捷宣言的签署人之一Alistair Cockburn，组织大家在2011年2月12日重聚，共同庆祝这个事件。这次聚会的目的是要回答以下三个问题。第一，在软件或者产品开发领域，我们已经解决了哪些问题（从而还有哪些亟待解决）？第二，哪些问题无法从根本上得到解决（因此我们不必再试图解决它们）？第三，我们能够合理地应付哪些问题——我们能够通过金钱、努力或者革新缓解的问题（因此，这些是我们接下来所要关注的问题）？

同样也是纪念这一事件，发源于美国的敏捷联盟组织（Agile Alliance）在2010年发起敏捷宣言翻译项目，旨在藉此将敏捷宣言翻译为不同的语言版本，将世界各地的敏捷爱好者、实践者联结在一起。目前已经有30个语言的版本，另外还有19个语言的版本正在翻译中。所有翻译项目，均发自于当地的敏捷社区，由社区自组织进行翻译。翻译的内容包括敏捷宣言的4个价值和12条法则。

2010年5月开始启动中文版的翻译，很快有关“简体”和“繁体”的争论变得很热，考虑到语言习惯以及社区物理上的区隔，6月份时正式分作两个版本同时进行翻译，徐毅协调简体中文版，麦天志和Chris Tong协调繁体中文版。虽然需要翻译的文字不多，但争议和讨论持续不断。除了社区，还有一些其他行业的热心人士贡献了他们的力量，最终简体中文版在2011年3月28日得以发布。

敏捷十年，超越十年

今年的敏捷大会，带着“敏捷宣言”签署十周年的纪念意义，以“Back to the Future　10 Years of Agility and Beyond”为主题（这个主题的意义太深刻了，还是不要翻译成中文的好），一方面回顾这些已经被实践证明正确的理论，另一方面审视我们如何将其思想发扬光大。

大会选题委员会从提交的超过1000个演讲主题中，挑选出250个演讲主题，组成17个分会场主题，超过1600人参加了大会。来自世界各地的敏捷实践人士，几乎将20多层高的Grand America Hotel和马路对过的Little America Hotel预定一空，很多人在年初1~2月份就订好了机票和酒店。

大会的主题围绕着敏捷实施和转型、商业与项目管理、嵌入式系统开发中的敏捷应用、教练与指导、团队协作与企业文化、开发语言 / 实践 / 技术、企业级敏捷、领导力、测

试与质量保证、用户体验与交互设计、客户沟通等。但在仅有的3个全体大会演讲中，2个与心理学有关，第3个则是从技术和非技术的角度，谈代码的艺术。

心理学家Babara Fredrickson，讲述她在正面情绪领域的研究成果：正面情绪可以帮助我们转变，转变成一个更好的自我。主题适合每一个人。而在50岁拿到工学博士学位，却又花了另外的20年研究心理学的Linda Rising，更是从一个心理学和教育学结合的试验入手，阐述了敏捷思维在教育、组织管理、软件开发，以及自我成长中的力量。敏捷思想越来越多地和各种学科融合，凸显了它的广泛性。

曾经是世界上首位会讲中文的CST（认证Scrum培训师）的吕毅，在大会上用熟练的英文与到场的来自世界各地的敏捷实施者，做了关于“大型企业敏捷转型”的经验分享和工坊练习，启发人们用系统化思维的方法，去分析转型中遇到的如自组织与无序、PO与团队协同等各种问题，寻求提升的办法。

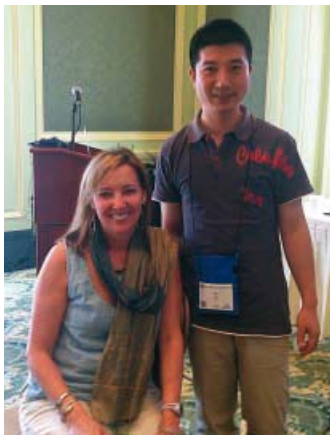


图3　来自Odd-e的吕毅和来自Rally的Jean Tabaka，共同组织了“大型企业敏捷转型”这个时段的工作坊练习

此外，大会的热门话题还少不了精益（Lean），包括精益基础的小游戏、精益工作流研究、精益创业等。

来自社区，服务社区

作为非盈利组织，敏捷联盟10年来一直在努力推动敏捷。今年的敏捷大会，更是把这一社区盛事推向高潮。本着来自社区、服务社区（From community and for community）的宗旨，以及实践者驱动（Practioner-driven）的原则，敏捷大会有很多不同于其他技术大会的特色。

Multi-model（**大会形式多样，日程安排灵活**）：大会除了安排主题演讲，还有很多包括工作坊、新兵训练、经验分享、动手实践、闪电演讲等形式，200多个时段，让很多人都能有机会分享他们的知识和实践。与会者也可以根据自己的时间和爱好，灵活选择自己喜欢的环节，定制自己的大会日程。

Social（**丰富的社交活动安排**）：对很多大会而言，除了大会演讲内容本身，吸引与会者参会和会后谈资论道的，就是大会的社交活动安排。本次大会的社交活动安排非常地丰富和有创造性。First Time Orientation帮助很多第一次来到大会的人，能够更加快速地融入到会议中，如何选课程、如何安排时间，很多问题在这个环节被问到。Open Jam是一个自由时段和自由空间的小规模聚会，让很多人可以在那里，自发地形成讨论小组，或是讨论某个主题，或是演示产品，或是宣讲，或是聚会，甚至来一场棋牌游戏！几乎在任何时候去Open Jam，那里都有人在讨论。Dinner with Stranger则帮助陌生的与会者结交更多的朋友，走廊的白板上贴上空白签到单，并提供了周边餐馆的就餐指导，在两天之内，这些白板上就签满了名字，陌生的人们利用这样的机会，相约到不同风味的餐馆就餐、聊天，很快就成了朋友。另外，大会上还有形式新颖的书市和拍卖，不由让人驻足 。

图4　Open Jam门前的白板上，被写满了与会者自行规划的日程

Creative（**大会在启发创造力方面做足了策划**）：在会议室和走廊，到处都可以看到白板、白纸夹、各种颜色的水笔；在会议室和走廊的墙上，到处都贴满了画着流程图或树形图的海报，以及各种颜色的即时贴。还有几位现场手绘者，记录着这十年敏捷的发展历史。所有的内容，都是参会者自发生成。大会赞助商更是充满想象力，遥控飞机、筹码游戏甚至萨克斯，都被搬到了展台。作为最大的“引导师”

图5 Rally的展台，现场手绘的敏捷十年发展历程

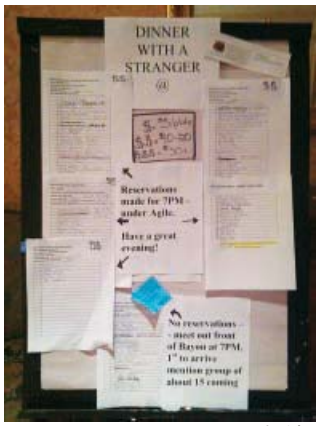


图6 Dinner with Stranger，帮助与会者结交朋友

图7 走廊里张贴的敏捷发展技术树

（Facilitator），大会组织者为与会者提供了自由发挥创造力的空间。

Fun（大会不仅是供与会者学习和社交的，还是供与会者娱乐和放松的）：很少有技术会议，提供给与会者丰富的茶水、零食、饮料、餐点及酒，让与会者不愿离开会场。在大会晚会上的精彩的跳床表演，以及乐队现场伴奏和演唱，更是将与会者的热情High到了极点。午餐中，还有拉斯维加斯的魔术师，亲临你的桌前，表演参与度极高的扑克魔术。更值得一提的是，大会中很多课程，都穿插有各种游戏、漫画、视频和讨论。观众不仅是在学习，更是在享受这些课程带给他们的快乐！

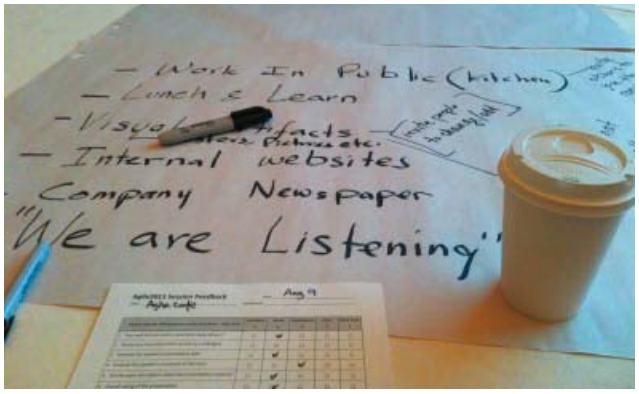


图8 随处都可以看到白纸和笔，随处都可以记录下你的思考

图9 Open Jam会议室，竟然提供了各种棋牌，还有积木

图10 拉斯维加斯的魔术师亲临你的饭桌前：我是来娱乐你的

方法学，价值观

敏捷运动并不反对方法学，实际上，我们大多数人在试图为方法学正名。我们希望能保持一种平衡。我们乐于建模，但不是写出设计图丢在满布灰尘的公司库房。我们支持写文档，但不是那些从不维护并很少用到的长篇大论。我们做计划，但要认识到在难以控制的环境中计划的局限性。有些人给XP、Scrum或任何一种敏捷方法学的支持者们打上“黑客”的标签，他们表现出的只是对方法学和黑客本意的无知。

敏捷运动并不局限于软件开发领域，实际上，我们大多数人已经在试图讲敏捷的思想推广到很多其他领域，包括商业，包括教育，包括心理，包括自我管理。一些人在质疑我们在“拿着锤子到处钉钉子”，但敏捷的价值观，以及一套系统化思考方式，的确在很多领域，正在被有效地应用。

盐湖城敏捷大会的闭幕演讲，Linda Rising的Agile Mindset，把整个大会带入一个高潮，并画上了圆满的句号，1600多位与会者起立鼓掌，长达几分钟。Linda更是将敏捷推向了更广阔的领域和更高的高度。最后，我们引用她的演讲：

敏捷的思维和心态，坚信我们所有人都一样，都在路上。

敏捷软件开发的流程也并非千古不变。随着我们对它的理解与日俱增，它也在变化中不断地成长。 幸运的是完美根本就无法企及，我们的旅途也不会有终点。

我们始终都会持续自我改进——就像我这样！

再试，再失败，更好地失败——萨缪尔·贝克特，爱尔兰诗人（1906年~1989年）。

把握技术革命的浪尖

腾讯副总裁、《浪潮之巅》作者吴军专访

记者 / 常政

“浪潮之巅”最初是2007年时在Google黑板报博客连载的一系列文章，因对世界IT产业高屋建瓴的视角、深入浅出的剖析，而获得盛誉。近期在读者们的千呼万唤下，《浪潮之巅》由出版公司Just-Pub正式出版，再次在IT圈内引起了广泛的关注。这是一本什么样的书？下面这段话可概括它的主旨。

“近一百多年来，总有一些公司很幸运地、有意识或者无意识地站在技术革命的浪尖之上。一旦处在了那个位置，即使不做任何事，也可以随着波浪顺顺当当地向前漂个十年甚至更长的时间。在这十几年间，它们代表着科技的浪潮，直到下一波浪潮的来临。”

不难看出，这是一部分析科技产业、商业机遇内在规律的IT历史书，难以想象的是它出自一位科学家之手——前Google资深研究员、现任腾讯副总裁的吴军老师。他是如何考量这部作品，并将其中的法则来解读我们身处的IT大时代的？最近，本刊记者有幸对吴军做了专访。

辨析时代浪潮的准则

记者：撰写《浪潮之巅》的内在动力是什么？

吴军：写这本书的主要目的，首先是帮助大家了解美国科技产业的发展。其次，我发现现在很多科技公司的投资者热衷于盲目跟风，不够专业，甚至连财报都懒得读，所以希望这本书能帮助大家对于IT投资有一个基本了解。另外，在美国时，经常会接待来美国访问的中国政府官员，发现他们对美国社会提的一些问题，存在普遍共性，大多和投资、产业、政策相关，说明国内普遍对西方科技产业感兴趣但并不熟悉，这也使我萌生了通过写点系统性的文字介绍一下的念头，好让大家无论是投资也好、制定政策也好，都少走弯路。当然最直接的原因是李开复希望我来写Google中国黑板报，他的邀请我不好意思拒绝。

记者：全书贯穿了一个主题，科技的发展轨迹是一个浪潮接着一个浪潮，只有顺应潮流才能成功。那么对于一个创业者，如何正确评判他是否处在时代的最新潮流当中？

吴军：新旧浪潮的更替，会遵循一定的标准。首先，这个浪潮不是泡沫，它能满足针对普通大众的很多需求。其次，是否具有关键的技术革新。比如PC热潮出来时，有两项技术（处理器、操作系统）是整个产业链绕不过去的。尽管硬盘、显示器等可以各式各样，却只有这两款技术主导了整个PC产业链。最后，这个产业必须是能盈利的，靠烧钱是维持不了长期繁荣的。

对于一个创业者来说，浪潮不会一夜间起，而后就马上落下来，一般都会延续十几到二十年，所以还是比较容易判断清楚。即使你做不了最早有眼光的人，跟着潮流做还是大有机会，比如像PC时代里跟Windows相关的软件行业，跟计算机相关的一些硬件行业等。记得当时中国的电子工业部，因觉得所有的计算机产品线都要补齐，要做小型机、工作站，结果却是小型机、工作站这些线基本失败了，而PC这条线倒是出了个没有太受政府扶植的联想，因为前者违背了浪潮。此外，顺应浪潮，最困难的是人的心魔，思维定式往往会使人不愿意承认新的潮流，结果就白白失去了很多的机会。

公司基因的力量

记者：在IT史上，不乏原来占据优势的大公司错失发展机遇的案例，但我觉得现在的大公司都学聪明了，一旦有新浪潮，纷纷厉兵秣马大量投入，这种情况下，实力相对弱小的创业者们还有机会吗？

吴军：我是这么来看的，大公司其实都有自己的基因，以中国最大的三家互联网公司，即阿里巴巴、百度和腾讯为例，你会发现它们虽然在其他领域也投入大量的资源，但真正做得最好的还是自己主营的领域。比如百度后来放弃了社区，阿里巴巴从收购雅虎中国开始就试图进入搜索领域，至今没有

《浪潮之巅》作者吴军认为，科技的发展是一个浪潮接着一个浪潮，把握技术革命的浪尖，可令你事半功倍。

做成。原因就在于它们的基因仅限于其所擅长的特定范围。再以国外的Google为例，在收购YouTube前，研发出了自己的Google视频服务（Google Video），但由于Google的主要精力还是花在搜索上，对在线视频的运营并不在行，加上投入度有问题，因为它也只是大公司里的一个小项目。这样，研发Google视频的员工们在激情和投入度上显然不能与完全专注于这个领域的YouTube公司相提并论。因此，尽管前者比YouTube早几个月上线，但后者很快就超过它，并把它远远地抛到了后面。所以，大公司重视一个新潮流，并不意味着就能把握住机会。此外，尤其是对那些超过万人的大公司，尽管领导讲话时习惯说“我们要万众一心”，可事实上万众一心是件非常难的事，所以一家大公司尽管会不断扩张、进军其他领域，但一般来说步伐会非常缓慢。

记者：说到公司基因，《浪潮之巅》还举了一些通过改变公司基因去迎合潮流、走向成功的公司案例，能否归纳一下，改变公司基因需要具备哪些条件？

吴军：公司基因，有的可改，有的不可改。可改的公司，往往是不断地把转基因作为一个强制性的任务。最典型的是3M，它有一个硬性的指标，要求每年的营业额必须有很高的比例来自于最近几年的技术——这样转型起来就比较容易。事实上3M公司最近的十几年，有35%的收入总是来源于五年之内开发的新品。但很多传统的大公司显然不是这样的，有收入的老部门习惯将没有收入的新部门卡死，比如原来的微软公司。所以要想成功地转基因：首先，公司的文化本身就是以转基因为核心，GE就是这样，一百年来经历了从做电灯泡开始，到无线电，最后到电视机的蜕变。其次，转基因也存在很多偶然性，成功与否看你能否把握住。比如美国的运通公司和美国第四大的富国银行（Wells Fargo，原意是威尔斯马车运输公司），很难想象，这一家信用卡公司和一家大银行，在历史上居然还是两家快递公司。因为过去的股票不是现在计算机上的一个数字，是实实在在的一张纸，当时运通公司和富国银行的前身就是给人传递股票、传递证券的快递公司，而通过传递股票、证券业务，使得它能够接触到买股票的有钱人，于是通过给这些有钱人提供买股票的贷款，最后变成了一家信用卡公司。当时美国买股票跟今天不一样，只要押10%，股票就可以先给你，所以运通公司和富国银行能通过贷款服务很容易地将这个生意做起来。当然在今天，这样的事根本不可能发生，因此有些时候，转基因需要靠一个特定的历史契机。

为什么选择腾讯

记者：《浪潮之巅》盘点了无数的杰出企业和领袖，您自己最欣赏哪一家公司和哪一个领袖？

吴军：从员工的体验来讲，Google是一家理想的公司。它非常善待员工，它说“不做恶”是真的落在行动上。两个创始人，尤其是拉里·佩奇，是个体面的绅士。它的问题是，一些员工被宠坏了。

从领导者的角度讲，我最欣赏GE的杰克·韦尔奇，因为使一家百年老店脱胎换骨、做成如今的规模其实是件很困难的事。世界上永远能找到几家这样的公司：如果没有了谷歌，可能有张歌、李歌，没有Facebook，也会诞生别的同类网站。但GE作为一家大公司，其拥有的每一家子公司都足以和行业的其他大公司竞争，而且快速发展了十几年，是件很难的事，但韦尔奇做到了，很了不起。在韦尔奇担任CEO期间，GE进行了上千次的并购和重组，保证了GE的活力并且能不断跟上时代的步伐。

另外，大公司通常官僚主义盛行，且每个行业里都有类似的潜规则，比如无论在美国还是中国的公司，如果你要越级向老板反映意见，是件很不敬的事。但杰克·韦尔奇却鼓励这样的行为，并随时会越下两级，约一个员工谈事或吃饭，这样便把整个公司的官僚作风，尽可能地降低。这对于一个大公司的领袖来说，是件非常不容易的事情。作为一个管理者韦尔奇可以用“国士无双”来形容。

记者：说到大企业的管理制度，谷歌自底向上的民主管理模式令世人称道，您觉得这样的模式能否在中国的企业或者创业团队里复制？

吴军：新创业团队肯定可以复制，因为大家目标明确、利益都绑在一起，即使有矛盾都能抛在一边，以工作为主。但

在中国的一些大公司内部，坦率讲是很难做到的，原因是以下三个方面：

第一，中国员工的社会成熟度普遍不够高。中国学生高考升学压力很大，基本大学毕业以前都是在象牙塔里生活。而美国的很多孩子，从小就很独立，基本上到了高中、大学，父母就不管了，都靠自己挣钱养活，其独立性和处理社会关系的能力很强。

第二，中国和西方的历史文化传承不同。中国古代习惯采用以皇帝为核心，各个省份设立总督这样的自上而下的体制，即使在今天采用的还是中央集权模式。而西方，从罗马帝国分裂以后，实现了真正的封建，就是一个骑士、贵族得到了封地后，就真的自己做主了，国王管辖的范围，超不过他的王城周边领域。大约公元1000年前左右，英国还出了个大宪章，大意说这些领主们和国王并不是一个完全上下属的关系，需要限制一下国王的权力等。所以西方产生民主的习惯不是一天两天的。

第三，中国的现代企业制度，也是最近30年才得到比较大的发展，在此之前每家工厂和政府部门的组织结构没什么差异。再往前追溯，就是胡雪岩那一辈开始办企业的时候，从兴盛到破产也不过几十年的历史。所以全部加起来，中国的企业发展进程可能不到百年的历史，远没有在全社会范围内，形成职业化、产业化的土壤，导致现在中国企业内的员工们职业意识普遍不强，所以一家中国企业如果贸然采用西方模式，不对员工加以管束的话，很容易搞得一团糟。

记者：海内外有那么多卓越企业，您现在为什么选择腾讯作为职业新起点？

吴军：中国现在的发展特征和二战后日本崛起的轨迹很类似，如果平移一下时间，相差40年左右，也就是说中国还有二十年的高速发展期。并且以中国的经济规模、发展趋势，她势必会孕育出像索尼、丰田这样真正意义上的大型跨国公司。当然历史永远不会简单地重复，记得英国最早期的大公司可能是机械公司，美国是电子公司，日本则是电器、汽车公司，而中国肯定不会重复它们的历程，同时一定会诞生这样一类科技公司，它会和互联网、IT技术紧密联系在一起。所以除了运营商外，我觉得在中国最有发展前景的就是腾讯、阿里巴巴、百度三家公司。但百度很难做到全球化，阿里巴巴则主要是以商业驱动，这都和我的个人价值观相差比较大。

为什么选择腾讯？尽管中国公司一般不适合象美国那样采用完全民主式的管理，但相对而言，腾讯公司的基因和文化，是最接近硅谷的。这有两方面原因：第一，公司的创始人，尤其是马化腾、张志东两人，都是工程师背景，和很多硅谷公司的创始人很像，他们在心态上和硅谷文化很接近；第二，腾讯总部设在深圳，深圳是一个非常开放的城市，所以腾讯相对其他中国公司更加外向，更容易包容海内外人才。考虑到这些因素，我觉得腾讯相对而言是不错的选择。

何时有下一个Google

记者：你在《浪潮之巅》里说了这么一句话，当中国不把房市、股市作为最快的挣钱手段时，就是诞生下一个Google的时候，能否预测一下这个时间大概是什么时候？

吴军：下一个Google肯定不是做搜索，甚至不一定在互联网上，但它一定是一家在世界上领头的创新公司，而且足够大。我还是拿日本与中国做对比，日本早期的时候也喜欢模仿，包括20世纪30年代中国抵制日货时，日货并不是高品质的代名词，它只是便宜，并且在70年代进入美国市场时同样如此，类似今天的中国制造。但进入90年代以后，日本孕育出了大量的新创造和新标准，比如卡式录音带、CD和蓝光DVD标准、混合动力车技术等。中国地域辽阔，聪明人众多，所以不必为它着急，一旦时机成熟了，一定会产生相应的成果。总得来说，再有20年时间中国应该可以产生自己的Google。

记者：听说您的下一本新书是《数学之美》，它和《浪潮之巅》有无一脉相承之处？

吴军：说到相同的地方，就是无论商业还是技术的内容，都遵循一个原则，就是要把复杂的问题简单化，让所有的老百姓都能读懂。任何一个事物都有它的规律性，《浪潮之巅》侧重科技和商业的规律性，《数学之美》则是阐述技术本身，看上去很玄乎，但背后也有一定的规律性，能用通俗易懂的方式表达出来。

结束语

一个月前，在北京贝塔咖啡馆举行的《浪潮之巅》新书发布会上，吴军有一番话令我印象深刻，大意是哪怕一个才华平平的人，如果赶上浪潮的顺风顺水，也能取得不俗的成绩。这说明人是一种社会性存在，世界宏观大势的流向与个人事业的命运存在紧密的辩证关系。而中国的软件工程师，特点是习惯于就某个技术领域精耕细作，鲜有“抬头看天”的意识，《浪潮之巅》无疑是一本帮助我们理解自身在IT产业内位置坐标和未来走向的指南，而吴军通过这本书所展现的广阔视野和深厚学养，也给大家树立了一个榜样。

怎样破解企业管理软件的困局

对话北京起步科技有限公司总裁马科

文 / 张祺

近年来，互联网、移动领域可以说是一片火热景象，相比之下，企业管理软件/信息化行业却面临重重危机：一方面，市场变化很快，电子商务、物联网等对传统行业有较大影响，用户需求加速变化；另一方面，管理软件厂商利润低，技术落后，人才流失严重，缺乏创新，难以满足用户的需求。行业内一些企业举步维艰，发展困难，甚至有一些企业已经关停并转。

怎样破解这样的困局？近日，《程序员》总编刘江与北京起步科技有限公司总裁马科进行了一次对话。起步科技从2000年开始打造企业管理软件的业务架构平台和建模工具，现在已经得到了众多客户企业的认可。

管理软件面临的难题

《程序员》：你当年是UCDOS的开发者之一，是软件业的老资格了。在2000年的时候怎么想到去做思维加速（起步科技的前身），转到企业软件，而且一开始就是做平台？

马科：主要还是看到管理软件存在很多问题。有问题，就意味着有需求，总要有人去解决。当时管理软件开发的重复工作量很多，效率很低。我们之前想做机器翻译。经过研究，我们得出一个结论，就是机器翻译最终要通过自然语言理解来实现，格式化的道路走不通。有了这方面的技术积累，我们进而发现，整个管理软件里面最本质的东西，其实就是一些描述性的信息，比如组织结构、业务流程等，这些是基本上不变的。只要能很好地表示这些知识，那么就有可能将企业管理软件开发中共性的东西抽取出来，做一个平台，降低开发的复杂性和工作量，提高灵活性。当然，做业务平台难度还是比较大的，这一做就是十年。

《程序员》：十年过去了，企业管理软件行业都有哪些变化？又面临哪些困难？

马科：企业管理软件这些年发展得比较慢，有些停滞了，困难很多，大部分还是一些老问题。

其一，行业整体混乱，低水平竞争，很像装修行业。就是不规范，而且没有质量保证。即使是有名的国际大企业、知名的国内企业，也有不少失败的案例，中小企业就更不用说了。

其二，开发效率低，周期长，应变能力差，往往计划赶不上变化，等软件开发出来，用户需求已经变了，始终无法满足。管理软件这几年，这种痛苦是存在的。用户的需求变得越来越快，因为急剧的社会变革，商业模式也得变。你看电子商务的兴起对零售业影响多大，原来我们买家电都是在百货商场里买，后来改到国美、苏宁了，现在呢，都上京东商城、当当、卓越了，十年迈了三大步。管理软件实际上是滞后于这种速度的，国内的主流管理软件厂商都远远跟不上客户的需求。你想，变的时候老板可能是在一夜之间想通了个问题，他恨不得明天就变。而我们管理软件呢，半年都出不来，一个项目可能动辄一年两年都没完成，在整个过程当中充满痛苦，客户痛苦，我们也痛苦。

其三，缺乏统一的标准，各个开发厂商开发的应用系统往往平台不一，数据格式不一，各自为政，各不相通，无法共享资源。举个极端的例子，某知名厂商甚至自己不同部门开发的产品之间也无法真正整合，只能做到界面上的假整合。而且现在用户界面还是C/S架构、只支持IE6和支持新浏览器同时并存，不同版本的软件无法兼容，这种不兼容使营销和维护流程变得繁琐复杂，最终用户得不到好的服务。

其四，整个行业缺乏创新，技术停滞了，落后了，工具也原始，相比其他领域的同行，危机深重。由于行业不景气，死气沉沉，逐渐失去吸引力，人才流失严重。

最后一点是近年来的新问题，我想着重讲一讲。我们有些技术人员习惯排斥新东西、新概念，总觉得阳光之下并无新事。其实这是不对的。本来企业用的软硬件，按道理应该是比普通老百姓个人用的要好，技术好，当然也贵。可是这几年，天翻地覆了。互联网就不说了，Facebook、微博比一般的企业软件好用多了，人家还是免费的。手机里苹果更恐怖。当初苹果推出手机的时候，很多人都觉得这个行业已经那么成熟了，会有很大机会吗？可是它的iPhone就一款产品，一年一个版本，到现在已经成为业界霸主。这种沧海桑

田足以让我们瞠目结舌。

所有的创新要站在一个更宏观更长远的地方来看，很多事情过了十年以后你再去看，会发现当时有些超前、不切实际的甚至可笑的观念，中间也有不少人因此成为先烈，但最终却引发了大变革。所以当别人有新概念时候不要一上来就批判，我们与西方同行相比，在对新事物的接受方面是有差距的，这个问题不解决，我们就很难成为领导者。

《程序员》：我曾经在《程序员》杂志卷首语写过一篇文章，叫“为什么buzzword很重要”。也是说新概念的重要性。历史上极限编程敏捷方法、Web 2.0、Ajax、SOA、云计算等等诸多时髦词汇，虽然有宣传和营销的意图在里面，但其内核是有技术上的真知灼见的，最后也都产生了推动技术和社会前进的力量。

马科：是这样的。企业管理软件同行必须认识到，现在我们最大的困难其实就是技术上已经落后于互联网了。最近几年互联网的兴起，网络游戏、移动相继起来，我们做企业管理软件的其实非常被动，最明显的体现就是优秀人才大量流失，行业里第一流的人已经不再关心企业信息化了。没有人才当然就没有创新，即使问题和需求在那里。此外，我们行业打交道的是大量的企业，而企业是机构，它们的决策和应变是比较慢的，因素很多，与消费市场个人决策只用顾及自己的体验和感受不同。企业里面很多东西很难改变，创新的难度也大得多。比如企业用户为了求稳，往往更信赖既有的品牌，即使它们做得并不好，不如一些新兴小企业的产品，但用户也不敢选。这也是很压抑创新的因素。

《程序员》：那么如何破解企业管理软件行业的这种困局呢？

马科：首先要有信心。我觉得现在是中国管理软件行业的一个机遇期。没有哪个国家这些年里新增了这么多企业，而且以高速在发展，市场环境、商业模式都在发生快速变化，相比西方社会的企业是在减少、衰退的，没有生气。此外，智能手机和平板电脑的迅速普及，正在改变终端用户和企业使用IT的图景。因此中国管理软件行业有可能创造一些前所未有的变革。

但是，要抓住这种机遇，必须以开放、积极的心态，拥抱变化，向互联网学习。

在技术方面，互联网和移动这些年积累的技术，其实已经为我们解决企业级应用中的难题打下了很好的基础，完全可以支撑管理软件进行变革了。比如UI层直接用HTML、CSS等Web技术描述，基础企业信息可以用现成的工作流和

马科，2006年创办北京起步科技有限公司，担任总裁。曾成功制定和实施多家企业的重大产业战略，并负责多个产品的大规模产业推广和市场运作。

语义标准描述。这是我们一直在等待的，也为起步科技现在推出X5业务平台和可执行业务建模工具打下了坚实基础。

我们所说的业务建模，是以业务描述、而非代码为核心来构建信息系统，使信息系统成为一种技术无关的描述性资源，在构建、发布和运行上具有技术无关性，从而提供了真正高效的开发、维护和管理模式，用户企业信息中心的技术人员，软件开发企业的研发团队，现在都可以通过这个平台和工具构建信息系统，实现高得多的效率和灵活性，满足多变的需求，使企业能实现随需而变、自我掌控。

而在商业模式方面，为什么我们管理软件行业不能使资源聚合化、平台化和公用化，提高复用性，像苹果那样建立一个企业软件应用商店作为渠道，更加个性化地为用户服务，重新打造全新的产业链呢？我想这是完全有可能的，关键在于行业形成共识，共同努力。

关于执行业务建模解决企业信息化难题的更多分享，请参见：http://www.justep.com/。

敏捷！

敏捷并不完美，一直在演进。对于敏捷我们没有现成的答案。

敏捷不是银弹，只是接受常识。单纯地依靠任何一种敏捷技术都不可能解决问题。

敏捷不反对方法学，相反努力为之正名。鼓吹敏捷以“传统”为敌，并不是真正拥护敏捷的人。

在本期封面报道中，我们邀请了不同软件开发理念的拥护者，以PMI、CMMI等不同的视角，探问敏捷的得失利弊，以UX、Tools、Testing等不同的维度，探问敏捷的发展与演进。在反复探问与比较中，虽然我们无法抵达问题的终点，但或许能找到一个适合自己的起点。

从敏捷的业务目标论软件开发

文 / 何勉

敏捷已成为软件开发领域的潮流，但单纯为迎合潮流去实施敏捷是不负责任的。开发方法和实践必须服务于业务成功，作为业务导向的敏捷实施成功的前提，首先必须问的问题是：通过敏捷实施要达成的业务目标是什么？为达成这些目标需要做到什么？如何做到？本文将从业务目标出发，分别从这三个方面展开讨论。

提高组织的响应能力

每一次软件产品的开发都是一个创造的过程，预知一切是不可能完成的任务。

首先，商业环境和市场的需求处于变化之中。Jonathan Rasmusson在*The Agile Samurai*一书中陈述了三个关于需求的简单事实：一、不可能在项目开始的时候收集到所有的需求；二、不管你收集到什么样的需求，它一定会发生变化；三、要做的功能，一定会超过时间和金钱允许的范围。加之，商业环境的不断变化，在项目的初始阶段固定所有需求，只是一个有害无益的幻觉。

其次，技术环境以及团队对技术的认识处于变化之中。软件产品的开发是一个探索和发现的过程。项目启动时，团队不可能建立对业务领域、遗留代码、实现技术、工具等方面的完整认知，处于对项目最无知的状态。随着开发的进展，团队不断积累关于业务和技术的知识，并做出相应调整，加之技术环境的不断变化，使得在项目的初始阶段就确定所有的技术方向和实现方案，既不合理也不经济。

不确定性是软件开发的固有属性。这既是一个挑战，也是一个机会。传统的管理和技术手段不再适用，不能把握不确定性，适应用户的需求，就会被市场抛弃；同时，不确定性也意味着无限的可能性，掌控不确定性就意味着把握了赢得市场竞争的机会。敏捷软件开发正是要构建组织适应不确定性的能力。《精益和敏捷开发大型应用指南》一书的作者Bas Vodde曾经这样定义敏捷软件开发的完美愿景——“构建组织能够随时在没有额外成本的前提下、交付产品或改变方向的能力”。在本文中，这被称作“响应能力”，并把它定义为：

响应能力——软件开发组织在尽量小的额外开销的前提下，通过随时交付，或改变方向来准确、即时地响应市场变化的能力。

“响应”是组织对外部（市场和用户）所做出的反应，具体包括：**交付产品**，当市场需要时向客户交付已完成的工作；**改变方向**，当市场需要，或组织策略发生变化时，做出针对性的调整。

“响应能力”指做出这种反应的能力，具体指标有：**即时**，在用户需要时随时做出反应；**准确**，应对市场的真实需求；**低额外开销**，不应该产生过多正常开发之外的成本，如返工和额外测试或牺牲产品质量等。

在网络社会，话语权越来越多地为用户所掌握，不确定性正在从事物背后的规律，变成被市场供需双方普遍接受的事实。面对不确定性，拒绝和避免它不再是一个选项，唯有打造适应它的响应能力，对于软件开发这类创造活动尤其如此。从业务角度来看，实施敏捷软件开发的首要目标正是要打造组织的响应能力。

改善协作

想象一个软件开发组织，它和市场紧密连接，可以随时交付完成的工作或调整方向，对市场做出准确的反应，这样的组织必然会在市场竞争中占据优势。响应能力是所有软件开发组织所期望具备的，但真正能做到的却很少。究竟是什么影响了组织的响应能力。从内部看，主要是堆积的“在制品”，和技术及学习“债务”。从外部看主要是和市场的连接程度，也即研发团队和业务人员、用户的协作。以下将分别分析这三个方面对组织响应能力的影响。

“在制品”是响应外部变化的负担。

所谓“在制品”（Work In Progress，WIP），是指已经开始但未完成（可以向最终用户交付）的工作。主要包括：未实施的决策、未实现的设计、未集成的编码和未验证的系统。

图1是一个典型的瀑布开发模型，需求阶段产出系统的全部需求规格说明；分析设计阶段产出整个系统的架构和详细设计；测试验证前完成所有的代码。这些都属于不可交付的“在制品”，在整个开发过程中，都无法交付软件以响应客户的需求。另外，在项目开发过程中，任何需求的变更，都意味着对“在制品”的返工，例如在编码结束时需求发生了变化，意味着前期的需求分析、架构和详细设计，以及编码都要进行返工，这引入了需求变更的额外成本。因为这些原因，团队无法在较低额外开销的情况下，随时向客户交付产品或改变方向，其响应能力必然是受限的。

如图2所示，理想的迭代模式下，“在制品”仅仅存在于迭代之内。团队在迭代开始时，计划一部分需求，迭代结束时产生可交付的软件，清空所有的“在制品”。这样的模式为提升组织的响应能力提供了可能，1）迭代结束时软件是可交付的，可以通过交付它们来响应市场的需要。2）每个迭代开始前，都可以对产品的功能需求和规划做出调整，响应市场、用户及相关各方的反馈。而且因为迭代结束时“在制品”的数量为零，不存在对“在制品”的返工成本。

即使是迭代开发，大部分团队并不能做到迭代产出物的直接可交付。如图3所示，在一个典型的迭代开发中，每个迭代产生可运行的软件。但可运行不等于可交付，由于技术手段等的限制，迭代结束时会有遗留未完成工作。随着迭代的进展，未完成工作不断累积，形成迭代开发模式下的“在制品”，它们损害了组织响应能力。首先，可运行的软件加上未完成的工作，才构成可交付的软件。也就是交付软件前必须完成这些工作，组织无法随时交付以响应客户的需求。其次，这些“在制品”也意味着风险，未完成工作的不确定性，使得交付更加不可控，损害响应能力。

技术和学习“债务”（debts）加大响应的难度。

“债务”是指将来某个时刻要偿还的负担。如图4所示，如果没有适当技术实践的支持，随着迭代的进行，既有代码的单元测试工作增加；功能回归测试工作量变大；代码质量因频繁变更而变差；系统越来越复杂，团队成员却缺乏对系统的理解。这些构成了软件开发中的“债务”，它们加大了将来系统修改、测试的难度，因而降低了系统的响应能力。

“债务”又可以分为技术“债务”（tech. Debts）和学习“债务”（learning Debts）。技术“债务”是指因系统的不良设计、实现和测试，导致系统在需要变更时，难以被理解，

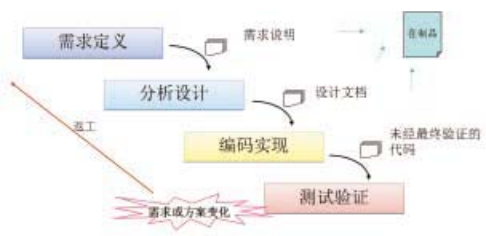


图1 瀑布模型下的在制品及其影响

难以改变；改变后，难以验证代码的正确性；难以对既有功能进行回归测试。学习“债务”是指，在开发过程中，团队成员未能及时分享和掌握业务及实现相关的知识，加大系统变难度，而且更容易引入质量问题，甚至新的技术“债务”。“债务”对系统的影响，并不会立刻显现。但长期来看，它会使系统响应变化时，耗费时间更长，成本更高，引入质量问题的可能性更大，严重损害组织的响应能力。

有效的协作决定响应及时性和准确。

“在制品”数量、“债务”水平决定了软件开发组织的内部响应性，与之同样重要的是响应的准确性。它取决于组织与外部的连接程度，也就是团队与用户及业务人员有效协作，以及团队内部有效协作，即时获取、整合市场信息，并准确传递至开发团队，有效地落实到开发活动当中。

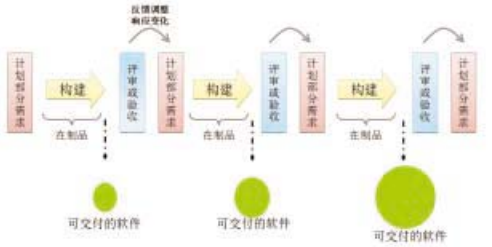


图2 理想的迭代开发模型

综上所述，只有降低“在制品”，维持低“负债”，并合理协作，才能保证准确、即时地响应，达成目标。敏捷开发的实践正是在构建这样一个系统。

实施敏捷

从一定程度上说，敏捷软件开发就是通过系统的实践，减少“在制品”数量，降低“债务”水平和改善协作来提高组织的响应能力。表1列举常见的敏捷实践，并分析了它们对于以上三个方面的贡献。我们将这些实践分成了三类——管理实践、团队技术实践和个人技术实践，下面将分别对这些实践加以总结。

管理实践

■ **用户故事**：用户故事是端到端的，足够小需求单元。用户故事应该是具备用户价值和可交付的。它作为敏捷开发中计划的单位，为减少“在制品”提供了可能。同时，它是从用户的角度出发，以用户的语言描述需求，是用户、业务人员和开发团队沟通的起点和工具，改善了他们之间的协作。

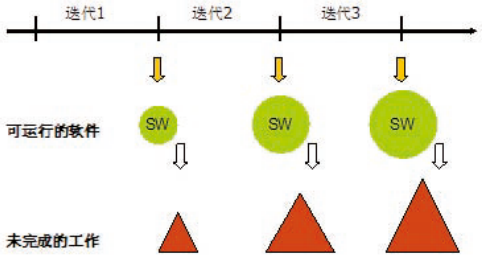


图3 现实迭代模型的WIP

■ **短迭代开发，小版本发布**：短迭代开发模式下，每个迭代产生可交付的软件，“在制品”始终控制在迭代内，处于较低的水平。小版本发布，即时获取用户的反馈，为调整产品的方向提供了最可靠的输入，使开发团队与用户的协作变得具体和有效。

■ **自组织的多功能团队**：多功能的团队，避免了工作在各个功能团队间的传递和等待，极大降低了“在制品”产生的可能，和维持的时间。同时，多功能团队成员间的知识共享和学习也降低了学习的“债务”。团队的自组织，使决策可以在团队层面迅速做出，减少了等待和批准的时间，避免“在制品”的累积。

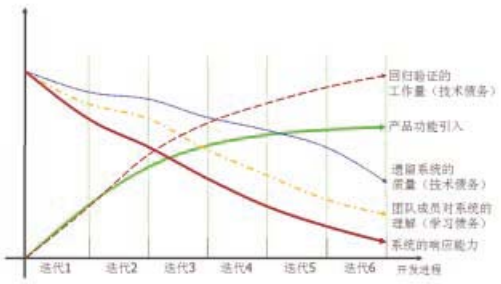


图4 技术债务和学习债务的积累

团队技术实践

■ **持续集成**：持续集成作为一个开发实践，强调小步地增加功能，并随时集成、验证，始终维持可运行的软件系统，使“在制品”数量控制在一个低水平。同时，持续集成作为一个团队纪律，保证被集成的代码能达到

定义的质量标准，如通过代码规则检查，自动测试等，屏蔽特定类型的技术“债务”。

■ **统一语言**：使用领域专家的语汇（领域语言）作为开发团队、业务人员以及用户之间的沟通语言，减少沟通中产生误解的可能，提高沟通效率。代码中出现的概念也应该与领域语言保持一致，这进一步确保了实现和领域之间的一致，既改善了合作，也降低了因系统理解难度而带来的技术“债务”。

表1 敏捷实践分类

分类	实践	英文翻译对照	WIPs ↓	Debts ↓	合作 ↑
管理实践	用户故事	User Story	++		++
	"短迭代开发/小版本发布"	"iterative development /Small Release"	++		++
	自组织的多功能团队		++		++
技术实践（团队）	"持续集成/持续交付"	"Continuous Integration /Continous Delivery"	++	++	
	统一语言	Ubiquitous Language			++
	接收测试驱动开发	"Acceptance Test Driven Development"		+	++
技术实践(个人)	整洁代码	Clean Code		++	
	设计原则和实践	"Design Principle & Practice"		++	
	持续重构	Refactory		++	
	"自动单元测试/测试驱动开发"	"Auto Unit Test /Test Driven Development"		++	
	简单设计	Simple Design		++	
	结对编程	Pair Programming		++	+
新实践	DevOps	DevOps	+		++
	AgileUX	AgileUX			++

■ **接收测试驱动开发**：在开发活动前，由业务、开发人员和测试人员共同参与，用实例对需求进行细化和澄清，确保大家的一致理解。这首先是一个良好的沟通工具，使三方的合作变得有序和有效。另一方面这些实例将成为功能测试自动化的起点，降低了测试相关的技术“债务”；这些实例同时还会成为理解系统的依据，降低了学习“债务”。

个人技术实践

■ **整洁代码**：代码仅实现功能是不够的，整洁的代码清晰的表达意图，易于理解和改变。整洁的代码降低了技术“债务”。

■ **设计原则和实践**：不管是Robert.C.Martin的S.O.L.I.D.原则，还是其它一些耳熟能详的设计原则，如消除重复、分离关注点和向稳定依赖等，其目的都是要产生一个高内聚、低耦合的系统，使系统易于变更和维护，减少技术“债务”。

■ **持续重构**：随着变更的引入，即使是好代码也有变坏的倾向，产生技术“债务”。持续重构正是要消除这些技术“债务”，维持代码的整洁，并使之符合基本的设计原则。持续重构应该是团队和个人的工程习惯，而非一次性的行为。

■ **简单设计**：过度设计带来的复杂性，本身就会成为未来变更的负担（债务）。另一方面，超越当下功能要求的设计无法得到充分的功能验证，会成为“在制品”。简单设计在实现功能、消除重复，以及清晰表达的前提下，力求用最少的元素实现系统。

■ **自动单元测试， 测试驱动开发**：单元测试是系统健壮性的基石，设计良好的自动化单元测试是系统最有效的防护网，它确保代码发生变更时，原有意图不被破坏。也只有有了单元测试的保护，重构才可能持续安全地进行。测试驱动开发则是产生良好设计的手段，同时它往往能生成最优和最及时的单元测试用例。

■ **结对编程**：结对编程有机综合了两个程序员的思维能力，更容易产生设计和实现良好的代码。同时，通过持续的代码检查，减少错误引入，并在开发人员之间有效传递了知识，降低了技术和学习“债务”。

以上，我们从响应能力出发，梳理了敏捷软件开发中的主要实践，这些实践并非为敏捷软件开发所专享。其实施，也非一蹴而就，需要以团队整体技术和管理水平的提高作为支撑。否则反而可能会带来新的“在制品”和“债务”，特别是“债务”。比如，引入单元测试自动化是为了减少技术“债务”，但如果单元测试自动化本身设计不良，测试用例过分依赖于实现细节。这样，对实现细节的改变，会影响到测试用例的正确运行，反而使单元测试成为负担，提高变更的成本，降低响应能力。

敏捷的实践之间是相互关联的，成为一个有机系统，才能发挥效益。**管理实践之间是相**

互支持的。例如，离开多功能的团队，开发任务就必须在多个功能团队之间传递，无法做到短迭代开发。**技术实践之间是相互支持的**。例如，没有单元测试自动化的保障，持续重构的风险将激增，在操作上变得不现实；有了持续重构保障，维持一个最简单的系统设计，需要时再通过重构，使设计适应新的变化要求，简单设计才更可能得以实施。**技术实践和管理实践也是相互支持的**。例如，短迭代开发中没有技术实践支持，就会不断积累技术“债务”，产品质量和开发效率不断下降；而，短迭代内全生命周期的反馈，也为各类技术实践的部署提供了便利。

综上，敏捷实践的应用和团队技术水平的提高，相辅相成，在应用中不断总结提高，构建和完善实践体系，加强客户、业务人员和团队的合作，以及团队内部的合作，持续、有效地减少“在制品”，以及技术和学习“债务”，提升组织的整体技术和管理水平，只有这样才能构建可持续的响应能力。

敏捷开发的最新实践

敏捷开发的思想并非在一夜之间产生，有些实践，如迭代开发几乎和软件开发的历史一样久远。上世纪90年代各类系统的敏捷开发框架和方法被相继提出，2001年敏捷宣言的发布，敏捷的概念正式形成，敏捷开发方法迅速普及。早期的敏捷实践，虽然也会涉及到业务和用户，但往往还是以开发团队为核心，重点关注团队内部实践。

随着敏捷实践的普及和完善，团队内部技术和管理实践作为组织响应能力的瓶颈被突破。近年来敏捷社区的关注重点更多地向团队前端和后端转移，一些新敏捷实践应运而生。

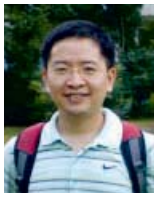
前端是指团队的输入，也即需求（包含用户体验）的获取、挖掘、澄清、整理和设计。得到比较多应用的新实践有SBE（Spec by Example）和 Agile U 。严格意义SBE和验收测试驱动开发（ATDD）属于同一实践的不同表述，它突出强调了，业务、开发和测试在前期通过表格化的实例对需求进行充分地沟通和协作，澄清和概念及一致性验证，并确保各方对业务和需求理解的一致。Agile U 把用户体验设计，更早和更紧密地融入敏捷软件开发过程，确保团队在迭代模式下，能持续交付具备良好用户体验的产品，同时加强与用户体验相关的响应能力。

后端是指团队的输出，也即产品的交付、运营和维护。得到比较多应用的新实践有持续交付和 DevOps。持续交付是持续集成的延伸，它的目标不仅是使软件始终处于可运行和通过适当验证的状态，它致力于确保整个开发周期中，软件都处于可交付的状态，从而降低交付风险，并缩减从概念形成到软件交付的时间周期。DevOps则进一步拓展，加强软件开发和IT运营之间的沟通、合作及整合，弥补两者之间的信息鸿沟，更快地交付软件产品，提供软件服务。

总结

在激烈竞争和充满无限可能的今天，技术、应用都在飞速发展，用户对体验的要求也前所未有的提高，响应变化的能力成为组织的核心生存能力。就这一点而言，敏捷对于软件开发组织是一个必然的选择，而非一个可有可无选项。

通过敏捷开发实践的系统运用，减少“在制品”数量、降低技术和学习“债务”，改善团队与用户及市场的协作，构建了组织的灵活响应能力。然而，敏捷对于软件开发，也绝非一个银弹，软件开发没有因之变得更加容易，甚至因响应性要求，团队和个人都面临更大的挑战。敏捷实践的正确实施，响应能力的建设，最终还是依赖与在实践中不断总结、提高，依赖于个人和团队总体能力的提升。



何勉

长期从事电信产品软件开发、产品开发项目管理和软件部门能力建设等工作。敏捷开发管理和工程方法的积极实践者和推广者。现致力于在上海贝尔全公司范围内推广敏捷软件开发。

“敏捷落地”路线图

文 / 袁斌

敏捷落地的基本思路

自敏捷宣言2001年发布以来，“敏捷”正在成为国内外软件企业热捧的词汇。但目前在国内实施敏捷的企业中普遍存在的问题是：敏捷如何在企业中真正落地，为企业带来实实在在的好处。敏捷提倡的价值观、原则和具体的实践，在企业内部会遇到各种各样的误解，外加团队的现实问题，导致敏捷在企业内无法落地，无法保证项目和产品的成功，团队可能最终放弃敏捷。Scrum是应用最广泛的敏捷开发方法。同时，它的失败率却非常高，其创始人之一Ken Schwaber估计75%尝试Scrum的组织无法获取他们预期的效果。

如何让敏捷落地？一个基本思路是：从总体上了解敏捷实践是一个持续改进的过程，在具体实践上从目前项目或者团队的最头疼的问题入手，在一个迭代周期内解决这个最头疼的问题，然后在本周期的回顾会议中找到下一个最头疼的问题，然后在下一个迭代周期再解决，以此类推……最头疼的问题的解决方案，以敏捷作为主线，结合PMBOK和CMMI等传统的管理模式。敏捷不是银弹，我们需要找到最适合本团队的解决方案。

敏捷实践的持续改进过程

敏捷实践是一个持续改进的过程，这里以Scrum为例说明。Scrum实践一般经历（如表1所示）的三个阶段。

表1 Scrum实践的三个阶段

阶段	本阶段内的特点
阶段 I	学习Scrum方法并尽可能严格按照Scrum的方法来实践
阶段 II	按照Scrum方法实践一段时间后，发现Scrum的理论和自己团队、企业现实有冲突，或者某些主题在Scrum提及但并没有深入阐述，需要在实践中探求解决方案，例如为什么第一个Scrum项目容易失败、敏捷团队、敏捷需求管理、大型团队的Scrum应用等
阶段III	实践Scrum方法一段时间后，发现很多Scrum中没有提及，但在实践中经常会遇到，同时需要解决的问题，例如如何做敏捷组织的管理者、如何成为优秀的Product Owner、Scrum在离岸开发、互联网产品等不同行业应用的特点、Scrum与CMMI / RUP等不同方法融合、对于Scrum解决不好的场景（例如事件驱动的项目）等

了解Scrum实践的三个阶段，可以让我们在实践过程中更容易发现团队所处阶段内的最头疼的问题，同时也可以避免“Scrum为什么有这么多问题”的急躁情绪。

如何解决具体的“最头疼的问题”

我们在这里分享一些在Scrum实践阶段I、阶段II和阶段III典型的“最头疼的问题”以及解决方案。

沟通效率不高

这个问题一般出现在Scrum实践阶段I。由于Scrum要求的是短周期迭代，通常是在2~4周做一次迭代，所以高效的沟通在每一次迭代中非常关键。团队在实践Scrum的初期，很多时候在学习Scrum推荐的会议形式，但会议的效率并不高。

这里以每日站立会议和回顾会议为例。经常遇到的问题是站立会议变成了汇报会议，每一个团队成员向Scrum Master汇报工作，同时对项目的整体风险和状态并不关心。回顾会议大家只是在迭代结束的时侯才开始回忆我们在整个迭代周期内哪些做得好，哪些需要改进，但在迭代周期中前期的时间内存在的问题一般都不能很好地被发现，因为时间已经比较长了，大家已经不能很好地描述当时问题的原因和背景，所以回顾会议并不能完全找到本次迭代需要改进的地方，那么在下一次迭代周期内这些问题还是会重复出现。

提高每日站立会议的效率，我们的解决方案之一是除了要每个团队成员描述三个问题：昨天做了什么、今天计划做什么以及我遇到的障碍之外，我们还会要求以下一些事情：昨天和今天做的事情要在白板上通过任务的移动表现出来；更新燃尽图；至少有一个成员会站在

整个迭代周期的高度来看截止到当日项目可能的风险；昨天成员提出的障碍，无论是否得到解决，一定要在今天更新障碍的处理状态。我们让每一个成员了解自己的工作对项目状态的影响，了解项目的状态和风险，让迭代周期内可能的风险在“今天”及早暴露和及时解决。

提高回顾会议的效率，我们的解决方案之一是在白板上留出一个区域，团队的任何成员，只要发现有需要改进的地方，或者认为某个方法非常好，就可以随时写在标签上并贴在白板上，做到“任何人有任何想法，可以在第一时间记录在白板上”，这样在回顾会议中可以很方便汇总整个迭代周期内所有成员的想法。同时，我们把初始拆分的任务、追加的任务以及Bug等用不同的颜色标签标注，在回顾会议中直接分析白板上的任务标签也可以直接得出很多有价值的数据。

因此，在我们的实践中，白板的使用在提高项目的透明度以及会议的效率上作用较大，这里和大家分享我们使用的白板（如图1所示）。

我们的白板是这样使用的。

- 白板中任务标签包括三种：黄色标签代表Sprint计划会议中拆分的任务，红色标签代表在迭代周期内新增加的任务，绿色标签代表测试发现的Bug。每一个标签都会写明任务名称、领取人员以及估算完成时间。
- 白板的右上角是燃尽图。
- 白板右下角用在搜集团队成员在迭代周期内的所有想法，无论是认为好的实践，还是认为不好的实践。这里有两个目标：首先在回顾会议中可以很清晰地了解团队在迭代周期内真实的感受，更重要的是我们可以在不好的味道刚刚出现的时侯，第一时间寻找解决方案，而不是等到回顾会议中再去分析、解决。

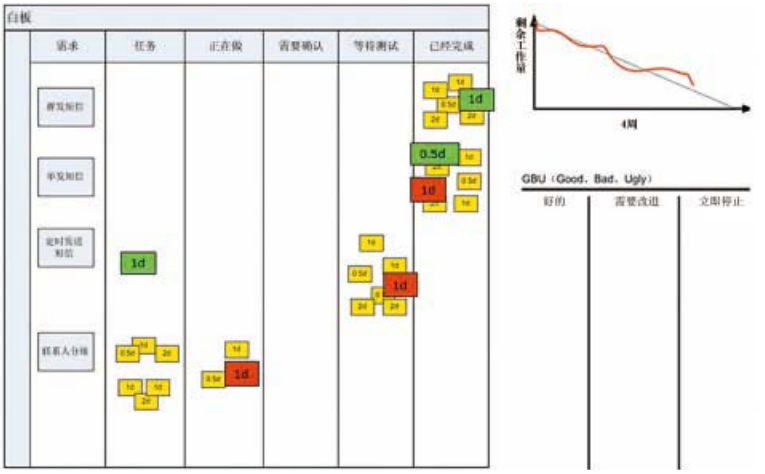


图1　工作中的白板使用

如何在部门推广Scrum

这个问题一般出现在Scrum实践阶段II。当部门的某一个项目组实施Scrum获得一定收益的时侯，部门会考虑在部门多个项目组实践Scrum。但和单个项目组实践不同，此时需要总结出适合本部门普遍现状的工作方式，同时也需要培养一些核心实施人员在其他项目组中起到推动的作用。

我们推荐下面这一种做法。

一、成立一个转型小组，这个小组由部门主管、其他项目组的主要负责人以及已经实施Scrum项目组（试点团队）的关键成员组成，最好再有一个外部的敏捷教练做指导。转型小组负责把实施Scrum项目组的经验和教训转化成部门实施Scrum的第一个工作版本。

二、试点团队在这个工作版本上继续迭代实践Scrum，转型小组可以旁观的身份适当参加试点小组的各种具体实践（例如计划会议、每日例会等），在回顾会议后试点小组把本次迭代的经验和教训反馈给转型小组，转型小组共同讨论后形成部门实施Scrum的第二个工作版本。

重复第二步骤，直到转型小组认为可以在部门实施为止。

具体的流程见图2。

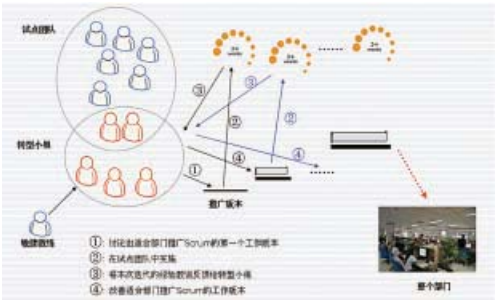


图2　Scrum推广

敏捷团队的绩效考核

这个问题一般会出现在阶段III。管理层从管理团队转变到领导、影响团队，绩效考核也会根据敏捷团队的特点和商业价值的追逐目标发生变化。在Scrum实践的阶段III，管理层会试图通过绩效考核等方式来影响团队的行为，逐步形成整个团队对最后交付负责的氛围，让团队可以自发消除技术债务，提高开发的效率，持续提升自我管理模式。

如何建立敏捷团队的绩效考核方式？这里分享一下我们对于敏捷团队绩效考核的考量因素的解决方案。

我们推荐敏捷团队绩效考核的考量因素分为两个层面：团队和个人。团队的考量因素包括“交付被接受”和“持续提高”；个人的考量因素包括“质量”、“工作量”、“帮助团队”、“主动性”和“成长性”。每一个因素可以像下面这样来度量。

“交付被接受”：Product Owner认可团队实现了迭代目标。

“持续提高”：每一次迭代的需求完成速度（Velocity）在提升，需求完成速度指一次迭代完成可交付需求的故事点或者理想日。

“质量”：测试人员发现的Bug数量反映了程序员开发的质量，外部客户反馈的Bug数量反映了测试人员的质量。

“工作量”：完成需求的故事点或者理想日。

“帮助团队”：用360度考评方法。

“主动性”：完成的需求质量，即完成的需求和Product Owner要求之间的偏差，参见“质量”。

“成长性”：“质量”+“工作量”+“帮助团队”+“主动性”的综合，主要适用于工作经验较少的团队成员。

在实践中组织要根据实际情况决定团队和个人两个部分的比重，同时也要考量团队和个人部分中各个因素的权重。敏捷团队的绩效考核不能是一成不变的，在年初制定的方案，需要根据团队和组织的实际情况做跟进调整，可以参见图3。

以下是组织在不同场景下的选择案例。

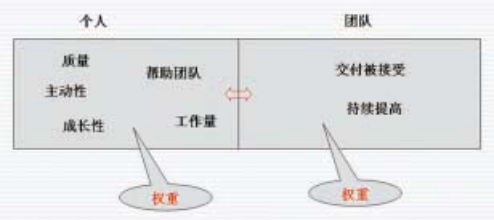


图3 绩效考核的权重

- 在实践Scrum的初期，个人的比重和团队的比重各占50%，同时个人部分中“质量”和“工作量”的权重较高，团队中“交付被接受”的权重较高。
- 对于团队中的初级开发人员，“成长性”的权重是最高的。
- 随着实践Scrum的时间增长，要逐步树立团队自我管理和整个团队对最终交付负责的氛围，此时个人的比重逐步降低，团队的比重逐步提高。在个人的部分，“质量”和“工作量”权重相对实践Scrum初期下降，“主动性”和“工作量”的权重相对实践Scrum初期时要提高。

总结

我对敏捷的理解是：通过快速反馈和持续改进来最大化满足客户，从而实现组织的商业目标。我对“落地敏捷”的理解是：在持续改进的每一个改进周期内解决最头疼的问题，解决的方案可以是敏捷的思路，也可以是敏捷和传统方法的结合。“对敏捷的实践一揽子全盘接受”和“实践敏捷，就是要对传统方法的彻底变革”都会不利于“敏捷”在组织和团队的落地。

注：（本文图表来源于迅思威尔敏捷咨询和开发团队的实践）

袁斌

迅思威尔科技有限公司（AgileDo）资深敏捷咨询顾问。离岸外包、互联网产品等多个行业的敏捷实践者。

敏捷测试的思考和新发展

文 / 朱少民

2010年为《程序员》杂志写了一篇《敏捷测试的方法和实践》，我们可以回过头来，看看过去的一年，敏捷测试发生了哪些变化。首先，我做了一个实验，分别打开2010年和2011年的“STAREAST Conference at-a-Glance”，输入Agile，2010年显示10个结果，而2011年显示17个结果，有一个很大的增长，说明敏捷测试越来越引起大家的关注。这只是一个表面的现象，我们还需要真正了解发生了哪些实质性的变化。

举一个例子，《敏捷测试：测试人员和测试团队的实践指南》的作者Lisa Crispin在StarEast 2011上有一个演讲——*Agile Testing: After the First Year, What's Next?* 其中提到，我们从传统开发方法转向敏捷方法，由于开发人员掌握了测试驱动开发（TDD，Test Driven Development），而测试人员部分地实现了验收测试和回归测试的自动化，所以我们活下来了，但我们在接下来深入实施敏捷测试时，还会面临新的挑战，甚至要克服更大的困难。当测试人员有了一年的经验，并拥有了敏捷方法的价值观、原则和实践之后，我们还不得不考虑如何不断改进持续的发布、如何有效地管理技术债务、如何对客户的需求有更好的理解，这就要求我们掌握更深的敏捷测试技术，例如将“精益（Lean）方法”用于改进敏捷测试的绩效，以及重构自动化测试的设计或脚本以提高投入产出比。

TDD向ATDD、BDD转化？

以前人们谈到敏捷方法，就会谈到TDD或UTDD（Unit TDD），但是究竟有多少个公司在采用TDD方法来写代码？而在采用TDD开发方法的公司中，又有多少程序员在全面使用TDD方法呢？TDD是一个纠结的问题。一方面，TDD的确是一个好东西，先写测试用例、后写代码，保证程序员第一次就把代码写对，也彻底解决了代码的可测试性问题，在代码层次上把缺陷的预防做到淋漓尽致。另一方面，多数项目很紧张，不可能给程序员足够时间去实施TDD，程序员对实现有极大兴趣，而对测试缺乏兴趣，多数程序员也不愿意或不会主动去做TDD。这样，TDD实践还存在较大困难，有比较多的争议。我看到一位作者写道：组里头TDD说了3年，据我所知，看完两本TDD名著，并坚持写单元测试的人只有我一个（我组里有开发人员15名）。

为了解决TDD实施不力，在过去一年，越来越多的人关注ATDD，即验收测试驱动开发（Acceptance Test Driven Development）。从2003年开始，人们逐渐实践TDD，而ATDD是在2007年Lasse Koskela写了一本书《测试驱动：Java开发人员的TDD和ATDD》，才开始引起大家的更多关注。从那时算起也有四年了，但在国内，则是最近一两年的事。当然，我们可以将TDD和ATDD结合起来使用，形成一种混合的方法模型。TDD和ATDD之间的关系，可以用图1来描述。

接着，BDD（行为驱动开发，Behavior Driven Development）也开始大行其道，那BDD是不是“做得比较好的TDD”呢？概念越来越多，概念的界限就难以确定，BDD可以看成ATDD的延伸，只是BDD更强调用户的视角、用户的行为，为ATDD注入了“Given，When，Then”这样特定的需求描述语言。2009年，BDD创始人在伦敦发表的“敏捷规格、BDD和极限测试交流”中，对BDD给出了如下定义：

BDD是第二代的、由外及内的、基于拉动的（pull）、多方利益相关者的

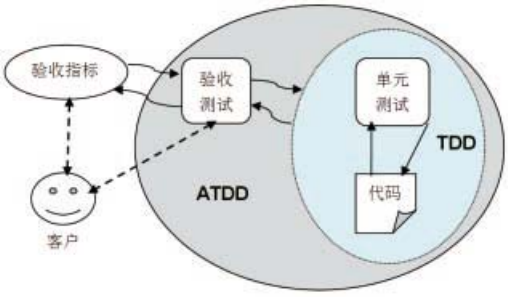


图1 TDD和ATDD之间的关系

（stakeholder）、多尺度的、高度自动化的敏捷方法。它描述了一个交互循环，可以具有带有良好定义的输出（即工作中交付的结果）：已测试过的软件。

但这个定义看起来还不够好，至少让我们明白起来还有一定的困难。实际上，BDD具有自己特定的“Given，When，Then”行为描述语言，和敏捷的user story极为吻合。所以“Given，When，Then” 行为描述语言才是BDD最显著的特征。

user story	行为描述语言
As a [X]	Given [Conditions /条件]
I want [Y]	When [事件发生：Actor+ Action /角色+行为]
so that [Z]	Then [observable results /可观察的结果]

TDD在写测试用例时，常常会提出“我们应该先测什么”，然后针对测试的条件来填充代码，而BDD则试图换一种方式去思考问题，即问自己“预期的行为是什么？”，可能会写出结构更好的代码。说到底，BDD更关注客户的需求，通过了解客户的不同行为，对客户的需求有更深刻的理解，从而借助对需求逐渐深入的理解来驱动软件开发。

TDD更重要的价值是其思想，就像传统的制造业，一定是先知道产品的质量标准或验收标准之后，才去设计、制造。从这个思想来看，TDD、ATDD和BDD都是一样的。不一样的是其具体的操作方法或实践，我们可以说，ATDD和BDD有一定的进步，但还没有到达完美的地步，还有提升的空间。在未来，首先就是如何灵活结合BDD、ATDD和TDD来构成一个测试体系，是一个发展方向；其次，就是在BDD、ATDD和TDD最根本的、共同的思想基础上，构成一个全新的、更完善的敏捷测试框架。后者的可能性更大。

探索式测试的地位

在过去一两年，在敏捷方法中探索式测试（Exploring Test，ET）也是一个热门话题，甚至有些人想用探索式测试来代替传统的用例测试（case-based test）或脚本测试（scripted test），走向另一个极端。探索式测试是对用例测试的补充，在非敏捷开发方法中也可以使用。只是在非敏捷开发方法中，有较为严格的需求规范和设计文档，有充分的时间去设计足够的测试用例，探索式测试只是作为一种辅助的手段发现一些隐藏很深的缺陷，并成为一种产品学习的工具以完善测试用例。然而，在敏捷测试中，由于迭代快、需求变化相对频繁，缺乏详细的需求描述文档和足够的设计描述文档，探索式测试发挥更大的作用，甚至在新功能测试中发挥决定性的作用。需要提醒的是，在敏捷测试中，回归测试应该仍然以用例测试为主，可以这样说，回归测试还是百分之百的用例测试。

探索式测试，实际早在1984年就由James Bach和Cem Kaner提出来，但为什么直到最近几年才比较热呢？这主要得益于敏捷开发方法的兴起，而敏捷开发方法的兴起又得益于互联网应用的迅速扩张。大家都知道，互联网应用越来越普遍，竞争越来越激烈，迫切要求互联网应用产品发布要快，再加上许多互联网产品的开发，都极具创新性、摸着石头过河，其需求不明确，要求开发周期短，频繁发布新的版本，及时获得市场和用户的反馈，不断修正以更好地满足用户的需求。针对被测对象，所掌握的信息不够充分的情况下，探索式测试就是一种很有效的测试方法。而且，把测试过程写下来（脚本化）需要时间，在敏捷测试中，时间显得更为珍贵。如果需求变化快，脚本化的测试用例维护成本也过高、甚至是极大的浪费。探索式测试的倡议者还认为，测试执行过程应该是智力活动的过程，这一过程越善于思考、越流畅，我们越有机会发现缺陷。而用例测试方法，有太多的停顿、不够流畅，会破坏

这一过程。

在敏捷测试中，当我们没有清晰的可参照文档、没有机会创建测试，我们自然会采用探索式测试。在James A.Whittaker 的《探索式软件测试》出版之后，探索式测试再次被推向

- 卖点测试法
- 地标测试法
- 极限测试法
- 深巷测试法
- 强迫症测试法
- 通宵测试法
- 破坏测试法
- 收藏家测试法
- 超模测试法
- 配角测试法
- 取消测试法
- 混票测试法

高潮，人们觉得有更多成熟的探索方法可以使用，例如：

但探索式测试缺乏良好的系统性、复用性，而且有些探索执行最终被证明是没有价值的。而我们关注有价值的探索式测试，将它们记录下来，使之成为固定的测试用例，用于将来的（后继的迭代周期）回归测试。回归测试验证已有功能是否正常运行，需要良好的系统性和很高的覆盖率，确保发布产品的质量，而且回归测试是不断重复的，在极有限的时间内完成越来越多的测试任务，这需要自动化执行、高效率执行回归测试。而这一切，依赖于相对稳定的测试用例。概括起来，敏捷测试可以看成：

新功能的（手动）探索式测试 + 脚本化（基于测试用例的）自动回归测试

敏捷测试的自动化

没有自动化，就没有持续集成，也就没有敏捷。在敏捷测试中自动化测试就更加迫切，这一点比较容易理解，每个迭代（如Scrum中的Sprint）都在增加新的功能，而迭代周期的时间相对固定，随着时间的推移，已实现的功能越来越多，这就要求越来越多的回归测试在时间相对固定的周期内完成。如果没有自动化测试，这是不可能完成的任务。

在过去一年中，敏捷测试的自动化又发生了哪些变化？如何重构自动化测试脚本以提高产出投入比（ROI）？下面就简单讨论一下敏捷自动化测试框架和敏捷测试工具等内容。

敏捷测试对测试工具要求简单、实用，随时可用，而对敏捷测试来说，自动化测试框架更为重要，它将负责集成各种测试工具，包括单元测试工具和验收测试工具等，还负责与持续集成、缺陷管理系统等整个开发环境集成。作为敏捷测试的自动化框架，一般会选择轻量型、开放类型的框架。说到这种类型的框架，可以参考RobotFramework（http://code.google.com/p/robotframework/）。在最近一年，其版本发布比较频繁，也日渐成熟。RobotFramework是基于Python开发的、可扩展的框架，所以适用于多种接口的复杂软件（如用户接口、命令行、Web Service、编程接口等）的测试。适合敏捷测试的框架还有Thoughtworks Mingle + Cruise + Twist，它能帮助测试人员和开发人员敏捷项目管理和协同工作、持续集成、测试自动化，允许使用BDD开发模式和Groovy动态语言来编写测试脚本，包括手动和自动方式来创建可复用的自动化测试脚本，并结合测试领域特定语言（DSL）实现自动化测试。无论是RobotFramework，还是Twist，它们都支持Selenium 2.0，这也反映了Selenium在敏捷自动化测试中的重要地位。当然，敏捷测试也可以采用类似Selenium 2.0+ WebDriver +PushtoTest那样的组合框架。

敏捷测试工具很多，但对敏捷测试来说，我们更要关注能够适应ATDD或BDD的测试工具，如Cucumber、RSpec、NBehave /CBehave / JBehave、EasyB、JDave等。也可以结合先前熟悉的测试工具开展工作，例如用自己熟悉的WatiN来结合SpecFlow 完成BDD模式的自动化测试。采用传统的微软Visual Studio也是可以的，因为在其 2010版本中，增强了对敏捷测试的支持，包括：

- 发布了Scrum流程模板Visual Studio Scrum 1.0；
- 支持“测试优先”的开发，支持ATDD；
- TDD的插件TestDriven.NET。

敏捷测试管理

基于敏捷测试的管理，更多体现了基于需

求测试和基于风险测试的平衡。对于新功能测试，不仅采用探索式测试，还要考虑基于需求的测试方法，借助类似BenderRBT这样的工具，进行需求的因果分析，建立其判定表并进行优化，从而建立非常高效的测试用例，使敏捷测试跟上开发的节奏成为可能。但整个测试周期，包括跨迭代周期的回归测试，都需要对测试风险进行有效的评估，在效率和质量上达到平衡，以保证所发布的产品的质量。

敏捷测试的管理，一定不要急躁、不要急于求成，要循序渐进获得改进，特别是从相对传统的测试方法转型到敏捷测试的团队来说，更要逐渐转型，如同敏捷方法本身所追求的“小步快跑”式迭代，这种转型本身也应被视为迭代过程，而不是突然某个早上，一切都变了。在敏捷测试管理中，不要试图通过一个迭代解决所碰到的各种问题，而是一个迭代只解决一两个问题，随着时间的推移，踏踏实实地、逐步地解决各个问题，即进入一个良性的循环，最终解决各种问题，使团队转型成功，无论在测试效率和质量上获得质的飞跃。

在敏捷测试管理中，尽管有比较多的原则要支持，例如“以人为本、为客户创造价值、面对面的沟通、简单化、响应变化和享受乐趣”等，但最重要的是以下几个方面。

- 持续的质量反馈：在整个开发过程中，持续关注质量，关注用户需求，发现任何阶段性成果的问题，持续向产品经理、开发人员等提供质量反馈。
- 持续改进测试方法，不断学习新方法和提高测试技术能力，不仅和开发人员保持技术同步，而且团队成员能力保持同步成长，想方设法把工作做到极致。
- 让团队具有很高的自我组织能力，每个成员都积极主动工作，自己能够解决自己的问题。

让我们享受敏捷测试的乐趣，享受成功！

朱少民

网迅（中国）软件有限公司资深QA总监。中国科技大学软件学院教学指导委员会委员，中国软件测试认证委员会（CSTQB）资深专家。在软件工程领域颇有建树，先后获得多项科技进步奖、出版十多部著作和高校精品教材，如《全程软件测试》、《软件测试方法和技术》等。

敏捷和工具

■ 文 / 蔡煜

敏捷软件开发绝不再是一个新名词了，但理解还是时时有偏差。敏捷宣言中的第一条“个体和互动**高于**流程和工具”，有人就误读为“有了沟通，一切都迎刃而解”，因此花费大量精力整顿团队合作，却轻视了工具（技术）。其实宣言中的意思只是想强调个人和沟通更重要而已。

实际上，既然是软件开发，在所难免得面临工具的选择，而且很多优秀软件实践离开强有力的工具支持都玩不转。在如今的软件开发世界中，工具（这里谈的是软件工具）层出不穷，数不胜数，那么到底该怎么去选择适合的工具呢？

本文将根据我十几年的企业级软件开发经验给出一点建议，和大家一起来探讨敏捷和工具（特别是在企业实施中的工具）这个话题。

为了避免不必要的麻烦，文中尽量用开源软件作为介绍，但这并不是说我排斥商业软件，恰恰相反，在很多时候，只有商业软件才提供了你需要的功能（当然大部分情况下开源软件会很快迎头赶上）。

背景知识：应用程序生命周期管理

聊到软件开发中的工具，一般都会提到这个术语“应用程序生命周期管理（Application Lifecycle Management，ALM）”，说句老实话，有点烂，谁都想把自己往上靠，谁都有自己的一套说法，图1为我心中贯穿企业开发项目全程的ALM全局观。

图1中列出了ALM中的各个子系统，以及我略有研究的相对应工具的名称，让我们一起先来简单地过一遍整个过程。

产品开发始于一定的需求，需求有好几个层次，从产品需求到项目需求，进而产生出用户故事（User Story），然后团队会分解出任务

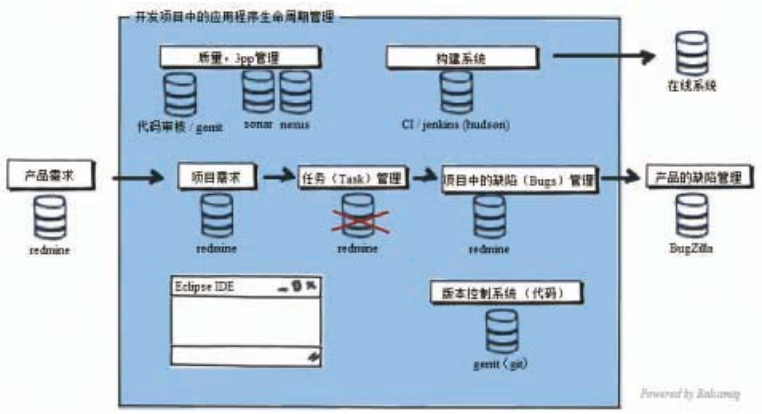


图1 开发项目中应用程序生命周期管理及其工具

（Task）。团队开发者利用IDE（如Eclipse）去完成相应的代码，单元测试完成后，再推送到代码库（git），这些和软件配置管理（Software Configuration Management，SCM）相关。

构建系统会从代码库中获取最新的代码，进行编译、打包、功能测试和系统测试，可以设置在质量系统中显示一些相关信息，如果一切顺利，甚至能直接上传到在线系统更新上线。此外不管是开发中还是运行中一般还都会有一个缺陷管理系统。

BugZilla大家应该很熟悉了，用Redmine的人少一些，但它实际上是一个非常灵活的项目管理系统，国内也有越来越多的公司在使用了，Nexus是一个Java软件包的管理工具，需要和Maven结合使用，Sonar是一个Java项目的质量控制工具，它集成了如单元测试、覆盖率、静态质量检查等内容。为什么任务管理下的Redmine有红叉呢？我们稍后会解释。

限于篇幅，本文只会谈到其中的一部分工具。作为参考，可以阅读Manning出版社的新书《Agile ALM》，它对SCM、单元测试、功能测试等话题进行了更深入的探讨。

敏捷中有些工具是必需的

如果你说敏捷实施大半年了，但持续集成(Continuous Integration，CI）服务器却还没有架起来，那我实在是不晓得你都在敏捷些啥东西了。无论你究竟“信仰”哪种敏捷，产品开发各个方面的自动化（Automation）和持续集成都必不可少。要了解持续集成，可以看看Martin Fowler的文章，可谓白皮书级别的经典，有中文翻译版。

使用CI就一定会用到CI服务器，可选的有很多，其中Jenkins是现在最流行的一个，而且架设起来也很方便。它是从Hudson继承而来（自从2011年5月Oracle决定把Hudson捐献给Eclipse组织，两者的关系和将来的发展方向也可能带来更多变数）。

在Jenkins构建服务器中，可以定义任务（在Jenkins中叫job），以完成一些构建步骤（如签出代码、编译、各种类型的测试、打包等），它有极丰富的插件（plugin）资源作为支撑，可以来集成产品软件开发的各个要素，它把你所需要的一切都自动化起来。

在Jenkins构建服务器的首页，每个任务还都有一个天气图标表示其状态，非常直观，例如“太阳”就代表着一切正常（至少从构建结果来看）。它是产品项目开发的晴雨表，项目状态是否正常一目了然。不管是对Jenkins不了解还是想提高，我强烈推荐阅读John Ferguson Smart的Jenkins开源书：*Jenkins: The Definitive Guide*。

对敏捷来说，并没有“CI服务器要还是不要”一说，它就是必须的，一定要好好地实施。基于持续集成再往前走，就是持续交付（Continuous Delivery）了，这也是敏捷的一个新热点，其中加入了很多新元素（如自动化验收测试、持续部署等）。

别让工具牵着鼻子走

工具很重要，但会不会有些误区呢？当然有，我们一起来看个我经常碰到的一个例子。

讲到敏捷实施一直都会提到白板（Whiteboard）的使用，先得提醒大家，它也是一个“工具”，白板的其中一种用法叫任务白板，主要是提供给团队使用的。

图3就是常见的任务白板，团队的需求，一般是用户故事（User Story），被放在最左边一栏“NOT CHECKED OUT”，作为团队一个迭代的输入，然后分解成任务（Task）用报事贴（俗称黄贴）贴在白板上“CHECKED OUT”那一栏，并签上自己的名字，就算是领任务啦。当做完了一个任务就把它（黄贴）挪到下一栏“DONE! :o)”，代表做完了，最右边一般还会留出部分空间，用来记录进度条和注意事项来提醒团队一些重要的事情。

如果说Jenkins构建服务器是产品开发的晴雨表，那么任务白板就是团队开发的晴雨表。

一般一大早在每日的站立会议（Daily

图2 Jenkins项目自己的Jenkins构建服务器

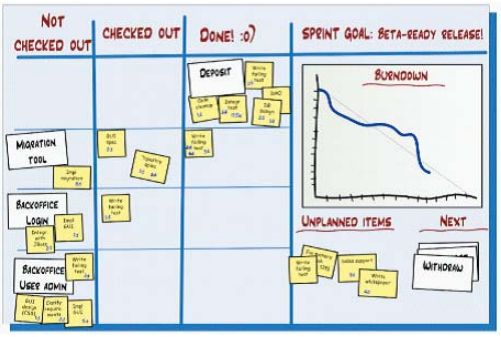


图3　每日例会用的任务白板（图来自《硝烟中的Scrum和XP》一书）

Standup Meeting）上，整个团队所有人都会围在白板前，分享所负责任务的进度，顺便挪动一下任务到相应的状态栏，用这种方式能够减少很多不必要的汇报型会议，而且团队成员也能很快地了解开发的整体状况。相信如果某个黄贴在白板上连着三天都没动，团队里一定会有人站出来帮忙的。（没有？！那还是先组织他们参加团队合作的培训吧。）

有时候团队坐得比较分散，或者有人喜欢流行的在家办公方式，这时可能会开始使用一些图形化的电子白板来管理团队任务，大家对着屏幕介绍进度，效果似乎还不错，其他人（包括经理们）也非常高兴，因为他们可以随时从网上了解到团队的进度了。慢慢地，面对面的时间看起来也可以节省下来，只要窝在座位上点点鼠标，挪挪电子黄贴（如果支持漂亮的拖拉就更好了）就已经足够了。

这是敏捷的好实践吗？

是否嗅到了一点怪味道？它就是我想谈的工具带来的误区，电子白板会削减团队之间的沟通，降低团队的透明度，违背了敏捷重视人和团队的原则。如果你的团队成员不经常去更新电子白板上的任务时间，慢慢地你看到的都是不太准确的信息了。有人可能说我们还是面对着电子白板开每日站立会议呀，那我希望你有个很大的屏幕吧（否则这样子效果会更差）。

那为什么没说它不对呢，因为在一些场合下，它还是有作用的。关键是团队要能充分认识到它的局限性，因此我极力反对在团队组建初期用电子白板，可以等团队充分领会到敏捷中人与人沟通的重要性后再引入。

这也是在图1中特意用红叉提醒不要用Redmine来管理任务（这个白板功能的插件也很差，因为没人用）。

所以要了解你的需求，不要让一些工具牵着鼻子走，要了解敏捷实施的目的，否则出了问题还以为工具不到位，拼命要更强的功能，结果陷入大误区。

有些工具会带来意想不到的好处

传统的敏捷著述中通常会提到CI的部分，然而工具的运用并不限于此，例如我在实际项目中用到的Gerrit，它非常契合敏捷的宗旨，也给我们带来了意想不到的好处。

代码审阅是一个不错的敏捷开发实践，让人非常头疼的是实施。大企业中通常是制定出一大堆相关的规范和流程来指导代码审阅。我听说好多公司都会把代码打印出来，进行走读，这可真是累啊。这种方式会给人不信任的感觉，而且应该是挺浪费时间的。

结对编程（Pair Programming）也是非常好的一种代码审阅的方式，值得好好地学一学，只是我对它并不太感冒，体会是在大企业实行结对编程的难度很大，当然如果能找到合适的推动者，那还是可以尝试尝试的。

那看看开源社区是怎么解决这个问题的呢，使用的又是什么工具呢？Google的Android系统是现在非常热门的开源项目，它的代码审阅（包括贡献者的代码）使用的是Gerrit，非常棒的东西，在自己的企业中架设起来也非常方便，我一用就爱上它了。

Gerrit是一个基于Web的代码评审和项目管理的工具，面向基于Git版本控制系统的项目，所以如果你没用git（干嘛不用呢），就没法用Gerrit了，接下来来看看在Gerrit中怎么实施代码评审的。

- 首先开发者（贡献者）的代码变更通过git命令被推送（push）到Gerrit管理下的Git版本库，推送的提交转化为一个一个的代码审核任务（见图4）。
- 代码审核者可以通过Web界面查看审核任务、代码变更，通过Web界面做出通过代码

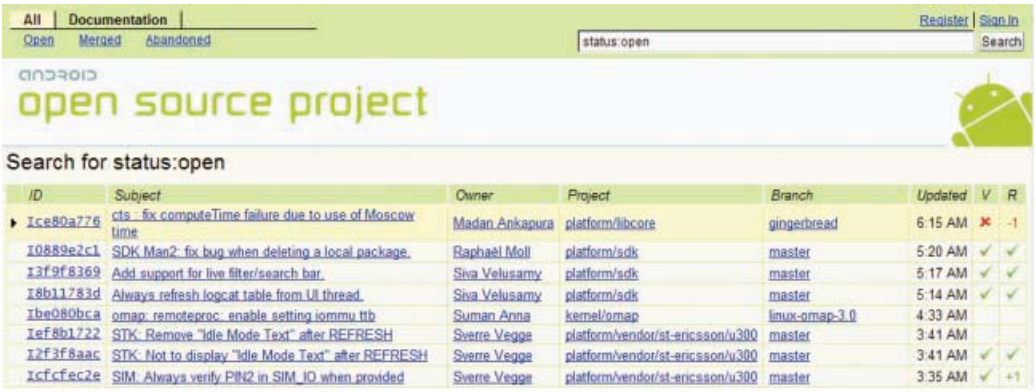


图4 Android的Gerrit代码评审系统截图

审核（Review）或者拒绝（Reject）等决定。

■ 测试者（一般可以设定为CI服务器执行）可以通过访问来获取（fetch）代码变更进行相应测试，如果测试通过，就可以把这个评审任务设置为校验通过（Verified）。

■ 最后经过了审核（Review）和校验（Verified）的代码变更可以通过Gerrit界面中提交动作合并到版本库的对应分支。

相比代码走读，它的好处在于，审阅动作发生在向主干提交代码前，可以只看变更的部分显得很贴心，网上随时随地审阅起来也很方便，这也是有别于结对编程的一个好处。

任何人都可以审阅提交的代码，整个团队的代码都一目了然，审阅起来更方便，非常符合开放、透明的敏捷精神。使用之后能够显著提高代码质量，甚至于等到习惯了以后，代码不被审阅一下，都觉得实在是不好意思提交代码到主干上去。

Gerrit中，通过特定分支，任何审核任务的代码变更都能访问，所以如果需要细看或是合并到本地都异常方便。

Gerrit刚开始实施可能会有点不习惯，毕竟整个工作流有所变化，因此实施时需要通盘考虑。

如上只是近期我特别喜欢的一个工具，其实类似的工具还有许许多多，这不过是沧海一粟而已。

总结

水能载舟，也能覆舟，工具和敏捷的关系亦如此，下面我给出几点建议：

■ 积极尝试新工具，如今的软件开发世界中，工具层出不穷，要不断与时俱进，了解最新动向和如何用有效的工具去辅助敏捷的实施，在自己的软件开发环境中去尝试和了解这些工具，尝试这一点很重要，不要只看说明或戴有色眼睛去看待这些工具，应该通过实践摸索找到最适合你的选择。

■ 人总是第一位的，要了解你的团队，了解他们的需求，有些时候甚至可以为了团队，冒险一下采用新技术、新工具，这样可以提高团队的凝聚力（敏捷要素之一）。我待过的一些部门推动Git时，就是这么来的，很多时候过多的讨论其优缺点反而是浪费时间，不如多花点时间多试试。

■ 实施敏捷也离不开组织的支持，特别大型企业涉及到的人很多（可能水平不一），这时候推动者就必须用专业的手段建立一个持续渐进的工具引入过程，这样会使得实施更容易些。

唠叨这么多，实际上也只是讲到一些皮毛，还有很多东西需要我们继续研究下去，本文得到了《Maven实战》作者许晓斌，敏捷宣言简体中文版翻译的协调者徐毅，敏捷之旅中国组织者滕振宇和同事代鹏的帮助，他们提供了很多宝贵的意见，特此致谢！

蔡煜

上海爱立信研发中心软件开发高级专家，作为软件实践的先行者，主要工作就是探索软件开发的最好最适合的方法和工具。开源、协作和敏捷的布道者。

PMO如何推动敏捷实践

文 / 蔡晓东

在带领大型项目团队攻坚时，我深深感到缺乏来自组织层面的支持。在大型项目中，干系人沟通、风险管理、采购管理的重要性凸显出来，而这些方面的问题往往并不是项目层面能够解决的。因此，当我成为PMO部门经理时，我发布部门的工作愿景：“建立科学、高效的项目管理体系，帮助项目经理成长并促进项目成功”。我相信，项目管理体系、项目经理成长与项目成功，这几个方面密切相关，而且都是PMO部门的工作目标。

PMO作为组织级项目管理机构，从项目管理角度对十多个项目团队、数十个项目进行管理。由于具体的项目情况千差万别，如若采用经验式的工作指导，很难让项目经理信服。因此，我们必须确定企业组织的项目管理方法和软件工程过程方法，并发布为组织的项目管理制度。同时，PMO作为项目管理制度的一个参与角色，与各个项目组一样处于制度框架的约束下，依据制度赋予的权力展开工作。这样，项目经理有足够的自主性，组织层面也能够有效切入项目活动，以支持项目成功。

因此，我们要选择一个项目层面的方法论，作为企业项目管理制度的主体。

作为“方法论”的Scrum

我们有很多互联网应用类和电信行业应用项目。在系统上线后，每过半个月或一两个月，还需要向用户发布一次新的版本。Scrum是最近一段时间非常火热的软件开发管理方法论之一。那么，Scrum能否作为“我们的”项目管理方法论，而在整个企业的范围内推广呢?

我们发现，一方面，Scrum的积极面在于激励开发人员对整个项目负责，通过自行认领任务，而获得了对任务完成工期和质量的承诺。另一方面，又通过在同行压力下的自主估算等活动，得出了一个相对客观的工时估算数据，以控制项目总成本。看起来开发人员和管理层都很满意。在某些特定条件下，我相信Scrum方法将带来项目成功。

然而，Scrum却存在以下缺陷，限制了它在更大范围内的适用性。

> Scrum社区有人指出，团队需要一个成熟的过程，我也相信这一点。可是，在中国的现状来说，一个工作小组由资深程序员带领两三名工作经验不足的程序员构成的情况并不罕见，我们不能等待项目团队逐渐成熟，积极作为才是负责任的PMO应有的态度。

“客户参与”可能导致软件需求规模严重超出预期

在网络游戏这样的Scrum案例中，客户越早参与越好，PO和开发团队可以识别合理的用户需求，并加入产品功能列表。而对于行业应用软件来说，客户是非常强势的。如果客户直接参与到团队中，与团队一起工作，日复一日，软件需求规模将迅速增加。同时，客户的投资预算则是相对固定的，不可能随需求的增加而同步增加。

因此，在行业应用软件开发的通常做法，并不能直接让客户参与到项目中，而必须将客

户作为干系人管理。需求文本必须经过用户正式或非正式的确认。Scrum对客户参与的设定，使得它在行业应用领域很难适用。

“团队自组织”对团队成员素质要求过高

Scrum团队没有真正的领导，任务工期估算和任务认领，都是由团队自主进行的。当开发团队具有相似的技能、相似的工作经验时，这种方法可以让员工不甘落后，勇于承担挑战。但如果工作小组由一名资深程序员和几位新程序员组成时，估算的数据就不再有意义，最后还是资深程序员领走最难的任务，新程序员领走简单任务。时日一长，大家都知道“自组织”只是形式，这位资深程序员就是小组的领导，他说了算。

Scrum社区有人指出，团队需要一个成熟的过程，我也相信这一点。可是，在中国的现状来说，一个工作小组由资深程序员带领两三名工作经验不足的程序员构成的情况并不罕见，我们不能等待项目团队逐渐成熟，积极作为才是负责任的PMO应有的态度。

缺乏对企业管理层的唯一负责人

“自组织”很美好，这是对开发团队成员来说的。似乎大家都没有了顶头上司，想怎么开发都可以自己做主了。然而，对于企业管理层来说，这个项目已经没有一个真正的负责人了。必须注意到，PO也不是Scrum团队的负责人，他无法对开发的进度和质量负责。那么，谁能够向企业管理层承诺，“这个项目将于某月某日交付”？从管理层的角度来看，有一个真正负责的人，是非常必要的！否则，一旦Scrum团队决定延迟交付，对于企业管理层来说，就意味着项目进度与成本的失控，而且也面临着客户投诉的压力。

成功难以在不同的项目团队间复制

基于“自组织”的设定，PMO及组织层面的其他管理者很难推动Scrum的推广。假设在某个项目团队中，Scrum取得了很大成功，那么，这个成功是否真的能够水平传播到其他所有项目团队中呢？我相信，优秀团队的优秀成员是有足够学习动力的，但这只是少数人。就我国软件行业的现状来说，需要推动力才愿意去改变的人才是多数。

综上所述，从企业管理层及组织级项目管理的角度看来，Scrum并不是一种非常成熟的方法论。纵观各种敏捷方法，也多有这样或那样的缺陷。这样吧，我们暂且把目光投向我们自己的实践经验，看看从中能发现些什么。

我们的实践经验

在几年前的一个大型电信行业应用项目中，我们遭遇了非常不理想的项目环境。

- 项目规模是公司有史以来最大的，管理层不知道该如何支持项目组工作。
- 客户方面没有愿意负责的人，尽管经过多次调研，但需求迟迟得不到用户确认。
- 项目团队是新组建的，团队成员没有共同合作过。
- 开发人员多为工作经验不足两年的员工，尤其缺少行业知识，对相关系统陌生。

在这样的背景下，我知道我们不能采用传统瀑布方式进行开发，因为风险太大了！针对该项目的主要问题，我们改变了传统的开发流程。

需求文档不是一次性完成的，而是先提供一个覆盖到各个功能点的Excel列表，形成项目的“任务清单”。后续根据开发人员的要求，需求人员分阶段与用户沟通，对需求进行逐步求精。

弱化概要设计活动。概要设计的时间被缩短到3天，只明确定义了子系统和大模块的功能，并定义了公共部分的基础库表结构。

强化风险管理活动。由于概要设计活动被弱化了，详细设计也没有充分开展，系统的很多细节未被充分揭示。因此，我把“风险”的概念做了最强的定义：凡是影响到任务如期交付的因素，都作为“风险”，项目成员必须将其明确加入到风险清单中。

短周期迭代。尽管系统是一次性上线的，但由于弱化了设计活动，开发人员是否正确理解了需求，是否确保了代码质量，是必须及时检验的。因此，我们设定了两三周为一个迭代周期。

流水线作业。需求人员根据开发人员的要求，在迭代周期T-1中就确认好了需求，并完善需求文档。开发人员在迭代周期T中开发完成，编译并自测通过后，将代码入配置库，通知测试人员。当所有开发人员的版本入库后，测试人员将配置库上打上版本标签后，并验证代码可编译后，迭代周期T到此结束。测试人员在迭代周期T+1中，将迭代周期T新开发的代码及修复的Bug着重测试，同时对所有已开发完成的功能进行回归测试。

我们选择两三周作为一个迭代周期，正好能够进行一轮完整测试。这样，在迭代周期T结束时，我可以确认这些工作任务已经“开发完成”，而在迭代周期T+1结束时，可以确认对应工作任务已经“测试通过”。

通过上述一系列措施，我们的项目团队很快走上正轨，尽管项目成员大多没有多少项目合作的经验，但很快就能自主沟通，不再需要我的参与。虽然我们没有实现“自组织”，项目经理仍然是团队中的绝对领导，但我们成功消除了沟通成本，没有因沟通不畅导致返工，或者让项目成员带着情绪投入工作。整个项目团队在一年多的项目周期中，共同为了项目目标而努力，并取得了成功。

在这个项目结束后，我发现，我们的项目实践与Scrum有很多雷同之处。值得一提的是，我们的实践方法更契合我们项目的实际情况；而将Scrum方法带入项目进行推演，却有很多问题难以解决。我相信，正如我们的方法来自我们的实践，适用我们的企业和项目环境，Scrum也必定是来自于特定的软件开发实践，适用于特定的企业和项目环境。

敏捷思想

有人说敏捷是思想，甚至还将敏捷思想延伸到教育、出版、管理、工业生产等领域。在我们的项目中，虽然没有得到任何一种敏捷思想的指导，但我们做出的实践，却非常“敏捷”：短周期迭代、需求逐步求精、高效沟通、团队自运转。我认为，如果说“敏捷”是思想，那么它并不是一种全新的、独特的思想，而是一系列有利于提高工作绩效思想的总和。在软件行业，敏捷思想更体现为以下几个方面。

敏捷包含了过程改进的思想。提起过程改进，我们最为熟悉的是CMMI。CMMI的过程改进，是以提高质量为终极目的，对过程的各个细节和工件质量进行精确控制，甚至定量控制（L4、L5），以达到控制最终产品质量的目标。敏捷则是把整个开发流程转变为短迭代方式，通过小增量、快速迭代，使得软件过程更加适应需求的频繁变化。

> PMO是企业变革的内驱力，它不但是垂直沟通的管道，也理应成为软件开发管理知识的会聚点。通过PMO的参与，敏捷实践经验将能够自下而上的总结汇总，再自上而下地推广落地。

敏捷包含了团队建设的思想。敏捷体现了团队建设的思想，主张通过持续地建设团队，使得团队成熟并能够自管理。我们知道，即使是中小型软件开发，“程序员”承担的也不仅仅是标准化“编码”工作，而需要他们积极发挥创造性，才能高效优质地完成任务。各种敏捷方法论普遍强调对团队的信任，以及对团队成长的期待，这有利于激发开发团队的积极性和创造性。

敏捷包含了技术与管理融合的思想。我们习惯于把技术与管理作为两个领域来看待，敏捷则指出了软件开发管理与技术之间的深刻联系。敏捷开发团队“自管理”，从而模糊了技术与管理的边界；而在自管理的过程中，又大量应用技术手段。一些敏捷实践，例如TDD，就同时包含了技术层面与管理层面的内容。

大多数优秀软件开发实践，都会触及以上三方面的思想。

敏捷则明确将其统合在一起，提出了一个高度融合的概念，这是非常积极的方面。我们不应把敏捷“神秘化”，成为一种只有“敏捷专家”才能深刻理解的东西。只有破除了“敏捷”的神秘色彩，才能客观地看待敏捷。我们应从提高项目绩效的角度，来衡量是否符合

“敏捷”思想。

我们需要什么样的方法论

当我们认识了敏捷思想是什么，就能够破除迷信，以实用、落地的态度来看待敏捷，并从敏捷实践中吸取营养。我们不必要非常系统地引入某种敏捷方法论，也不必奇怪为什么这种方法论无法良好运作。作为PMO，我们有能力自主推进敏捷，改善效率，控制质量。

首先，我们从自己的项目实践中起步，获取敏捷实践的种子。在一个企业组织中，不同项目团队的项目管理和软件工程水平有很大差异。PMO应深入到各项目中，总结优秀项目团队已经实施的敏捷实践。我们将发现，部分敏捷实践已经在大部分项目中实施，并被验证为对项目的效率与质量有显著效果。而另有一些实践还只在个别项目中实施。在我们的案例中，迭代已经被非常普遍地采用，很少有一次性把需求做完整的情况。而TDD还只是个别项目成员在关注学习。

其次，建立企业级项目流程规范，将经过验证的敏捷实践成文。如前所述，一些敏捷实践已经在大部分项目中广泛实施，且被验证为有利于项目绩效的提高。这样，这些敏捷实践以及相关的项目流程，就可以被固化为企业级项目流程规范的一部分。PMO将在项目流程规范中找到适合的切入点，代表组织层面干系人参与到项目。

有了企业级项目流程规范的保证，即使项目组遭遇人员大量流失，有了组织层面的干预，新的项目团队也不得不进行基本的项目流程和敏捷实践，并将感受到敏捷开发的效益。

第三，通过多种渠道建立企业组织中的信息沟通管道，鼓励新的敏捷实践。在企业级项目流程规范中体现的敏捷实践，只能是经过验证有效的一小部分，而且PMO的参与度仅限于适合组织层面干预的里程碑等时间窗口。一些充满生机活力的项目团队还想要尝试新的敏捷实践，让他们去做吧，我们看看结果。如果做得好，想要推广到其他项目组，依赖项目组到项目组的水平推广方式太低效了；而PMO又不能直接干预过多项目细节。但PMO可以成为企业的信息枢纽，通过简报、白板、项目总结会议等多种方式，使得企业组织中其他项目团队了解正在发生的事情，并自主决定是否导入这种敏捷实践。

当一个新的实践被三分之一的项目团队接受时，PMO就必须系统研究这个实践对于企业环境的适用性，并考虑是否将其作为企业级项目规范的一部分。从而更大力度推动，使得“惰性”较大的项目团队不得不“敏捷”起来。

最后，是否提高项目绩效是检验敏捷实践的重要标准。一些敏捷支持者提出，敏捷方法能够让团队自管理、能够让开发人员做主，让开发人员工作更开心。我必须明确指出，这些都只能是副产品，而不是敏捷的目标。当项目团队组成后，其责任就是在限定的工期和成本内，做出符合质量要求的软件产品或系统。这个责任，要么是直接由一个具体的人，即项目经理来承担，要么由整个项目团队共同承担。如果整个项目团队都不承担这个责任了，开发人员是开心了，管理层如何面对当初给客户的承诺?

对于打着“敏捷”旗号，实际上使得项目团队绩效下降、开发人员不重视本职工作而片面强调自管理等的“虚假”敏捷，PMO必须坚决干预，这既是对管理层负责、对项目成功负责，也是对相关的项目成员的职业生涯负责。

敏捷开发在企业组织的迅速推广，离不开组织级项目管理机构的参与。PMO是企业变革的内驱力，它不但是垂直沟通的管道，也理应成为软件开发管理知识的汇聚点。通过PMO的参与，敏捷实践经验将能够自下而上地总结汇总，再自上而下地推广落地。在今后的敏捷大潮中，PMO必将扮演日益重要的角色！

蔡晓东

PMBAR项目管理社区PMO领域专家，参与PMBAR组织的《IT项目管理那些事儿》一书创作。在电信行业IT企业中工作13年，担任过系统架构师、项目经理、产品线经理等职位，现任技术总监兼项目推进部（PMO）经理。

企鹅快跑——腾讯敏捷历程揭秘

文 / 艾永亮

腾讯这只企鹅在13年的成长历程中，不断长大，但却并不笨拙，这其中的秘密就在于研修了敏捷方法！本文就将为您揭开其中不为人知的敏捷故事。

天生敏捷基因

企鹅出生在极速变化的互联网行业，出生之时便面临着四大挑战。

海量用户的需求：企鹅服务于数以亿计的互联网用户，在保证业务稳定的前提下，更要满足海量用户不断变化的需求，因此企鹅必须要竭尽全力快速实现一个个新需求，如果采用传统的开发方法，用户是无法接受的。

行业的迅速变化：互联网上新概念、新玩法、新应用层出不穷，一会儿SNS、一会儿团购、一会儿微博，一步落后步步落后。

竞争对手的压力：虽然很多人都觉得企鹅很可怕，但是行业变化如此之快，企鹅再大再强也不可能把所有产品做到第一，取舍之间就有可能被其他公司超越，毕竟迫于竞争对手的压力。

自身发展的需要：企鹅希望能为用户打造一站式在线生活，让用户更加方便地在网上冲浪，但要想实现这个目标其实很难，需要做的产品太多太多，要完善的功能点太多太多，而资源又太少太少，急需一种高效的方法来支撑产品开发。

幼年时的企鹅虽然遇到了这些问题，但那时候它还不知道有敏捷方法的存在，但好在有几项与生俱来的小聪明，借此支撑了几年的发展，后来证明这几项小聪明其实都有着敏捷的影子，我们管它叫草根敏捷基因。

拥抱变化：从不拒绝变化，只要对用户有价值的，即使推倒重来，也要作出最有价值的功能给用户。

重视反馈：为了能够听到亿万用户的声音，建立许多反馈渠道，例如QQ群、Qbar、客服、意见反馈、内部反馈、用户CE等，借此收集用户对现有功能的意见和新功能的期望，进而指导产品经理的工作。

快速发布：很早就建立完整发布平台，可以非常快地发布到全国各地的服务器上，这使得企鹅具备了产品快速上线，缺陷快速修复的能力，目的也是为用户提供更好的服务。

快速改进：建立很完善的用户数据统计分析平台，用于发现影响用户使用的瓶颈，发现用户操作的习惯、发现对用户最有价值的功能，从而有的放矢进行产品优化，提升用户体验。

敏捷历程

小聪明毕竟也只能支持几年，因为业务发展实在太快，必须系统学习一种有效的方法来来支撑进一步的发展需要，经过多方打听，企鹅重要发现了一项绝技敏捷方法，经过多方学习，开始在内部有条不紊地尝试起来。总体来说敏捷学习分为三重境界，下面我们来共同回忆一下这段学习经历，每个阶段都采用了“点、线、面”相结合的学习方法，我们也将按此思路为大家展现。

中规中矩

刚刚开始学习，企鹅并不急于随意发挥，因为它认为只有真正理解敏捷的精髓，才可以自由运用，因此老老实实、中规中矩地练习起敏捷方法来。

点（标杆小项目+敏捷教练）：我们在公司

各个业务线选取了若干具有代表性的项目尝试敏捷。选取过程中主要遵循如下标准：

- 团队有需求，有明确的问题
- 团队愿意接受敏捷教练
- 重点项目，资源用在刀刃上
- 教练能力可以帮到团队
- 团队规模适中（5~12人）

接下来我们的专业敏捷教练会下到团队通过如图1的步骤开展敏捷实施工作。

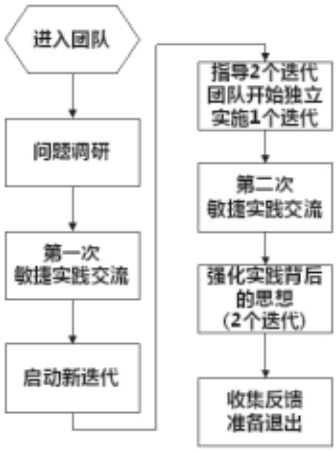


图1 敏捷实施流程图

其中实施过程主要分四个迭代展开，着重在如图2的六个方面进行指导。

图2 敏捷实施指导图

线（提炼模型）：经过近两年的深入实践，结合自身项目特色，我们将企鹅的敏捷提炼出来两种模型。这两种模型成为企鹅实践敏捷的基本套路，从“线”的角度为相似项目提供更具操作性的指导。如图3和图4，精炼地展现了两种模型的特色与实践。

面（培训+工具平台+敏捷研发奖）：“点”和“线”分别实现深入和升华，但是如何对更大范围的项目产生影响，必须通过“面”的手段广泛地传播敏捷思想和实践，为此我们也是通过三个方面开展。

首先是培训，我们结合多种敏捷方法、企鹅特色，开发出了多门敏捷相关课程，全方位地为公司员工进行培训，主要有一些系列课程，供大家参考。

表1 腾讯敏捷培训内部课程

序号	类别	课程名		课时(h)
1	基础类	敏捷开发新兵训练营		7
2		Scrum（敏捷项目管理框架）精要		3.5
3	实践类	管理实践	传统项目管理的转变-敏捷项目管理方法	3.5
4			建设自我组织的高效团队	3
			敏捷开发中的用户研究	2
			敏捷需求管理	2.5
5			TAPD工具应用解决方案	2
6		工程实践	灰度发布	2.5
7			持续集成-基于CI工具的实践	2
10			TDD+敏捷测试	2.5
11		产品案例case	QQMail：Mail是怎么炼成的	2
12			QQ会员：会员老树开新花	2
13			手机QQ：打造手机上的互联网世界——走近手机QQ	2
14			超级QQ：超级QQ——浴火重生	2
15			QQhummer：焦点的背后——探寻QQ研发模式	2
			Qzone：中国最大的在线时尚生活平台——亲密接触Qzone	2
17	管理类	精益管理者指南		2

其次，敏捷实践的固化与更加高效的运转需要强大的工具进行支撑。为此腾讯组建了一支团队专门开发了适合自身的敏捷产品开发平台Tencent Agile Product Development（TAPD）。它提供了敏捷产品开发全生命周期管理，包括产品管理、项目管理、发布、缺陷报表等。另外TAPD的强大之处还在于它内嵌了多项优秀的敏捷实践，如用户反馈、特性裂解、迭代计划、时间线、故事墙、燃烧图、发布计划等，并为不同业务类型提供多套整合解决方案，如Web应用、无线应用、游戏、桌面应用等。

最后，为了鼓励更多的项目积极尝试敏捷方法，我们通过“卓越敏捷研发奖”来鼓励积极实践并取得明显效果的项目。奖项评选主要从项目管理、迭代能力、CE（Customer Engagement）敏感度、团队经验分享等几个方面来衡量。截至目前，已经有15个团队获奖。

皆为我用

经过四年的苦心研修，可以说已经掌握敏捷方法的“形”，但是“神”还掌握得不够，于是开始新一轮的学习。希望借此让内部项目对敏捷的理解更进一步，达到融会贯通的程度。同样，腾讯也是从“点、线、面”三个方面来进行的。

点（部门级重点大项目+派驻项目经理）：要想融会贯通，必须从关注众多小项目转向关注部门级重大项目。因为这种项目由于规模较大，特别是常常涉及多个子项目协作，前一阶段的一些方法已经不太适用，急需寻求新的方法帮助此类项目提升效益和质量。

但是如何在此类项目中开展敏捷，其实谁都没有底，原先的教练已然不能轻松地指导项目了。为了更加深入了解项目，找到合适的解决方案，并有效推进方法的落地，采用了派驻项目经理的方式与项目团队合作。

通过这一阶段的实践，我们丰富了公司敏捷模型，增加一种“大象模型”，特别适用于部门级的大项目采用，详见图5。

线（TAM）：众多团队提出了敏捷指导需求，但是原来单对单的辅导效率太低，必须有一种方式可以让受益面更广，我们花了大约一年半的时间终于找到了一种非常有效的方法Tencent Agile Master（TAM）训练营。

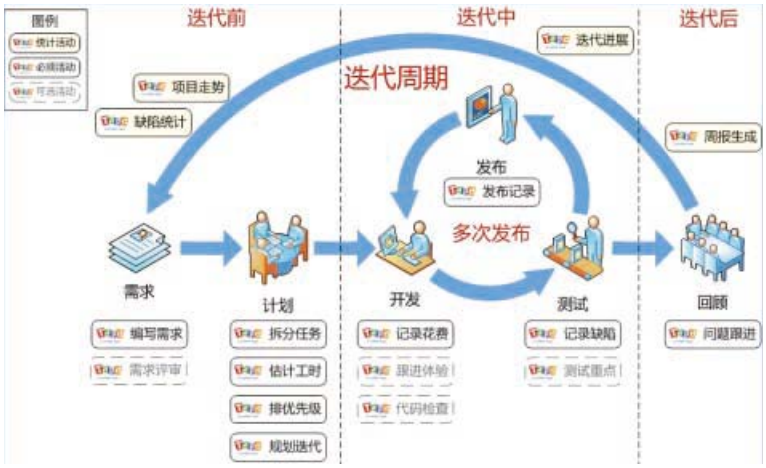


图3 极速模型

图4 迭代模型

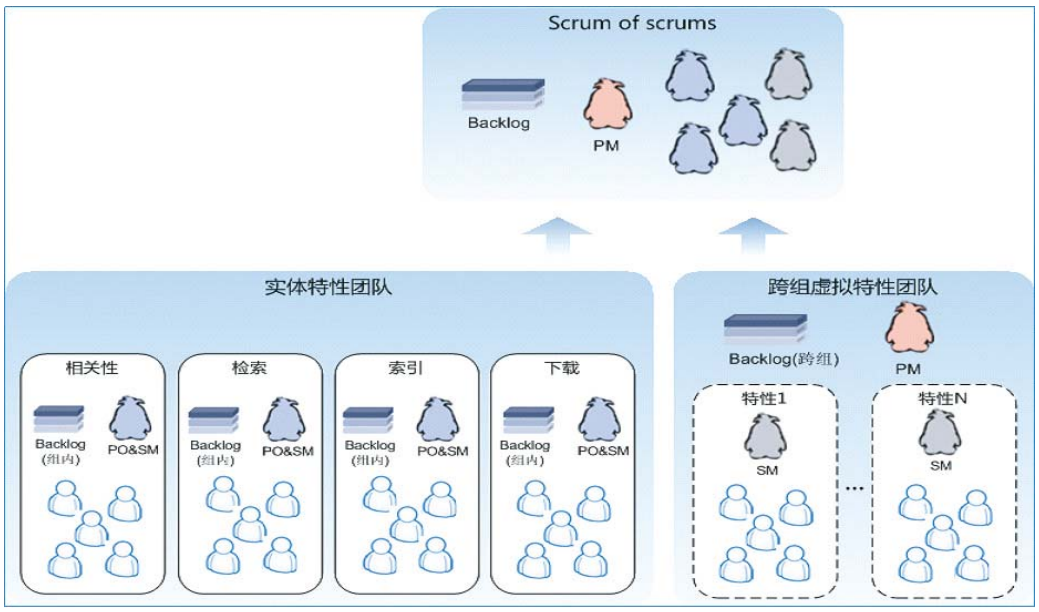


图5 大象模型

对象：有意愿实施敏捷的团队骨干成员，特别是项目经理，因为项目经理能从全局角度推动团队采用各项敏捷实践。

方式：Training + Action + Coaching + Sharing。Training部分涉及六门课程《敏捷项目管理基础》、《敏捷需求管理》、《敏捷规划》、《敏捷实施跟进》、《敏捷团队建设》、《质量与持续改进》。全过程分为四个迭代进行，前三个迭代每次集中进行两门课程的授课，接着要求学员在各自项目中切实实践，每门课程中设定3～5项实践点，期间有专职的敏捷教练进行指导和答疑，之后再对实践效果进行评估，在下一迭代开始时，请学员分享各自实践心得，并进行深入的探讨。第四迭代将对总体实施效果进行评估，进而对学员实施TAM认证，并请CTO为其颁发证书和奖品。

TAM在广度和深度两条线同时有效加速了敏捷的推广，取得了非常好的效果，值得向大家推荐。

面（敏捷俱乐部）：这一阶段的“面”更侧重于公司内部敏捷经验的交流和传播，因此我们发起了“敏捷俱乐部”这一组织，由专人负责运营，通过“线上+线下”结合的方式实现“敏捷知识管理+分享交流”。

线上活动主要在公司内部知识管理平台KM上展开，通过KM的文章、讨论、活动、问答、资料、季刊等形式，使得敏捷相关知识得以推广和沉淀，截至发稿前，线上敏捷俱乐部已经有500多篇文章。

线下活动主要包括公司内外专家演讲、专题讨论会、Open Party、模拟场景训练等活动，让大家相互认识，交流经验，探究答案。每次活动之后请大家在线上总结收获，使得线上、线下紧密结合，互为促进。

自成一派

腾讯经过两年进一步的研习，敏捷运用已经达到了随心所欲的程度，不过它还不满意，希望进一步发展，于是开始了第三阶段，目前尚在进行中，目标是形成自己的敏捷流派，并将敏捷方法发扬光大。

点（公司级重大项目+大项目经理）：随着公司不断发展，越来越多的公司级项目涌现，必须从公司层面跨业务的开展， 这对我们来说又是新的挑战，此时我们还是采用委派大项目经理的方式探索新的敏捷模式，这一工作目前还在进行中。我目前负责的一项涉及全公司的技术改进项目就是一个例子。

线（企鹅敏捷价值观）：基于三大模型，企鹅经历多年积累了产品价值观，我们希望能够将敏捷精神与公司本身精神相融合，形成企鹅特色的敏捷价值观。目前我们已经初步确定七大价值观：无快不破、海量之道、柔性可用、立体监控、灰度发布、产品微创新、体验文化，目前正在整理完善中。

面（开放共赢）：如今互联网迎来了开放大潮，敏捷方法也要开放，具体来说我们打算从内部和外部两个层面进行敏捷开放。内部开放是希望通过跨业务跨部门的轮岗实现内部敏捷方法和经验的实质性流动，并为大家提供更大的发展空间。外部开放是要走出去，将企鹅研修的心得回馈给业内，让所有公司能够共享敏捷实践成果，实现共赢。

结语

本文全局性地为大家展现了腾讯敏捷实践之路，以及未来的发展方向，实践证明敏捷是非常适合互联网开发的方法，但需要一些适应性调整，希望此文展现的一些具体实践能为正在尝试敏捷的公司提供一些借鉴。

艾永亮

腾讯公司敏捷教练&高级项目经理，曾参与QQ农牧场、Qzone商城、SOSO、无线应用、网络游戏等业务的项目管理与教练工作。有着多年敏捷实践和咨询经验。可通过腾讯微博（aland_ai）、新浪微博（alandai）与作者交流。

敏捷热点问题的多角度杂议

文 / 张克强

今年，敏捷软件开发的潮流继续冲击越来越多的组织，不同组织在导入或应用敏捷开发方法的过程中发出了各种各样的声音，几个突出的问题形成了讨论的热点。本文试图以多个角度来讨论一下这些问题。

测试驱动开发

在讨论测试驱动开发之前，先澄清一个问题：测试驱动开发是否包括验收测试驱动开发。测试驱动开发（Test Driven Development，TDD）存在两种理解：一、包括验收测试驱动开发（Acceptance Test Driven Develop，ATDD）在内，这个是广义的理解；二、TDD采用单元测试手段，主要针对非界面代码的，与用户故事或需求一般没有直接关联，这个是狭义的理解，TDD和ATDD是并列的。多数人认为应当采用后一种理解，因为TDD主要采用单元测试方法，典型工具有xUnit系列，ATDD主要采用界面自动化测试方法，典型工具有Selenium、QTP等；TDD主要用于设计和编码，ATDD主要用于需求分析和确认。下文TDD即是采用后一种理解。

在极限编程（Extreme Programming，XP）中TDD是与简单设计、重构、持续集成等紧密配合的，这些组成了一套威力强大的组合装备。TDD是其中最突出的外在表现，XP中TDD遵循“测试驱动开发金规”：先写一个会失败的测试，再写一个新特征，永远如此。对比在武侠世界，XP的TDD，极限测试驱动开发（eXtreme Test Driven Development，XTDD）属于神器级别，功力不到者是没法自如使用的，反而可能伤了自己。经典的单元测试方法、架构方法（比如常见的MVC）和设计方法（包括常用的设计模式）是开展TDD的基础，TDD的学习和实施是循序渐进的，由简入繁的，由浅入深的。所以把XTDD看成是第一个极端，把没有任何单元测试视为第二个极端，把经典软件工程中V模型（其

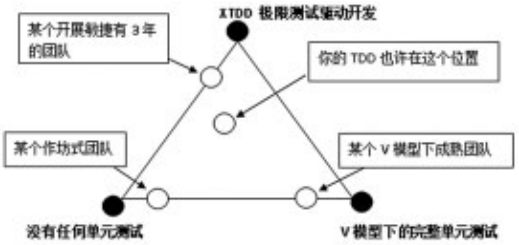


图1　TDD的三角关系

在单元测试方面的特征是针对已经有的功能代码编写单元测试，以保障代码的质量）的完整单元测试看成第三个极端，三者组合为一个三角型，如图1所示。从没有任何单元测试到XTDD，存在多种多样的中间状态，比如只对模块接口进行TDD，比如进行模块级的架构设计后开始TDD，比如在识别了主要类后再开始TDD，比如对个别有把握的模块先编码后加抽样的单元测试，等等。在这个三角型中选择一个合适的点，相信能够发挥单元测试和TDD的最佳效果。在没有足够功力之前，先不必开展XTDD。简单否定TDD是不恰当的。

从CMMI角度看，TDD能够满足CMMI3级中技术方案过程域（TS）、验证过程域（VER）和产品集成过程域（PI）的多个实践，是不错的工程实践。

从传统的瀑布型生命周期方法及其衍生的V模型的角度看，TDD下的基本设计比较简略，甚至没有，难以满足设计里程碑评审的要求，TDD没有详细设计，而单元测试各项指标（比如覆盖率、测试频度、测试用例数量等）却超过V模型，总体上违反了V模型。但如果不了解源自于V模型下的单元测试技术、面向对象设计技术，TDD也是难以开展。

从XP角度看，TDD应当开展到极限。

从Scrum来看，TDD本身不属于Scrum，应当由团队来决定是否采用TDD，如果是，也由团队来决定采用何种程度的TDD。

从ASD（Adaptive Software Development）、

FDD（Feature Driven Development）等其他敏捷方法流派来看，并没有明确要求，根据需要来选择开展TDD。

团队自组织

“自组织”是敏捷中难以把握的一个词，英文是self-organizing，也有地方是self-organization，还有self-organized，以上三个词的中文都译为“自组织”，从英文来看，显然三者是有区别的。

在敏捷软件开发宣言中没有提自组织，在敏捷原则中相关原文是：*The best architectures, requirements, and designs emerge from self-organizing teams*。译文：最好的架构、需求和设计出自自组织团队。Scrum中用的词也是self-organizing。从字面意思上理解，敏捷原则指出了自组织为方向，推荐自组织团队，但并没有强制要求必须自组织。这与现代管理学对待知识工作者的方向是完全一致的。已经有研究结果表明，基于X理论并且很好地适用在体力工作者上的管理方式是不适用在知识工作者的。

通过英文单词self-organizing和self-organized

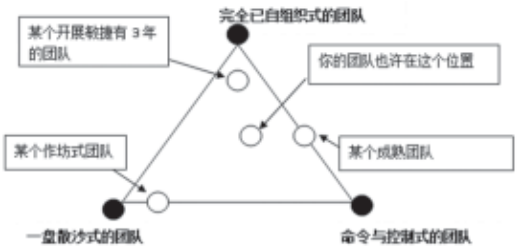


图2 团队类型的三角关系

可以看出，自组织可以分为两种状态，一种是正在自组织中，第二种是已经自组织了。self-organizing更像是一种趋势，当团队展现出向self-organized的趋势时，这个团队就可以算是自组织中的团队，简称自组织团队。

自组织的特征：“没有外部指令，系统按照相互默契的某种规则，各尽其责而又协调地自动形成有序结构。”所以没有团队内部领导者并不是自组织的本质特征，自组织团队内部不一定没有领导者。

在Scrum的团队方案中，取消了项目经理角色，给Scrum Master加了不少限制，团队内部没有指定的领导者，不允许给他人分配任务，如果团队能够有效运转的话，自组织程度必然很高，可以说超越了self-organizing，达到了self-organized。这样对Scrum团队就意味着必须强制采用自组织方式。对比XP的理念，好的东西就发挥到极致，Scrum有点这样的味道：自组织团队是好的，就要把自组织发挥到极致。一盘散沙、群龙无首的团队是个极端，自上而下命令与控制的团队也是个极端，完全已自组织（self-organized）团队也是个极端，存在一个如图2所示的团队类型三角型。现在已经有多个例子表明，没有团队领导者的Scrum团队是可以获得很高的团队绩效的，照样可以形成原来在个别杰出项目经理身上的团队领导力。当然也有例子说，标准Scrum团队搞不下去。

如果团队成员只会做相应报酬的工作量，还可能基本达不到其相应的报酬，大多数人都在混日子啊；如果给他们自由，他们会只会做他们感兴趣的事，反正不干正事，这个团队离图2所示的完全已自组织顶点还很远，要有效发挥团队能力的话，一个富有领导力的领导者是最可行、最现实的解决方案。

另据报道，当前这个世界上已经存在一家名为Semco的公司，这家公司的团队没有内部领导者，团队成员的薪酬已经内部公开，并且各团队成员能够心平气和协商确定各自的薪酬，团队绩效非常好。这样的团队真可以算是自组织的巅峰了。

我认为适合的才是最好的，应在三个极端构成的三角型中寻找合适的位置，把握三者之间的平衡，以团队贡献目标为导向，不必追求形式上的极端，所以并不赞成仅仅为了搞Scrum而取消团队内部领导者。从传统软件工程看，团队内部领导者的设立似乎是理所当然的。中国政府信产部推出了名为“信息系统集成项目经理”的资格证书。从CMMI来看，在CMMI for Dev 1.2中，明确给出了项目经理的名词解释：负责计划、指导、控制、构建、激励项目的人，项目经理对满足客户负责。显然的，这个解释就是传统软件工程中理所当然的解释。而在2010年10月发

布的CMMI for Dev 1.3中，项目经理的名词解释没有了，在正文中，说明了项目管理相关目标和实践，没有提项目经理。这个改变显然是敏捷所带来的。因此最新的CMMI for Dev看，只要满足项目管理相关目标，没有团队内部领导者的团队能够符合要求，当然传统的项目经理也能符合要求。Scrum关于自组织的高要求在前文已经说明，而从 P及其他敏捷方法流派来看，“自组织中”的团队是得到推荐的，而且自组织程度需要比较高，但缺省的仍然保留了项目经理或领导者的角色。在敏捷的另一个流派—精益（Lean）当中，就强调经理发挥好教练的作用，并且将此作为精益的基础。水晶方法集中存在项目协调者或项目经理的角色。

流程

敏捷软件开发宣言的第一个价值观指出“个体和互动高于流程和工具”。“流程”对应的英文是“Process”，在有些地方也译为“过程”，下文中“流程”和“过程”为同义词。由于敏捷宣言和原则总共篇幅不长，并没有完全说明其中问题，导致了如下有趣又矛盾的现象。

- 部分敏捷的执行者抱怨以Scrum为代表的敏捷流程比较生硬，让员工都是那一副副苦逼的脸，一些团队依然察觉到许多刻板、教条之处，比如站会，Sprint的干扰，一旦听到Scrum Master或团队试图“照着书本做Scrum”，就会为自己敲响警钟。
- 部分刚刚了解敏捷的人，尤其是团队以上的管理者，认为“敏捷不怎么讲过程规范，引入敏捷意味着冒较大的风险，为稳妥计，暂时先不搞敏捷了吧”。

敏捷软件开发宣言是在软件工程基础上提出来的，出发点所比对的是以瀑布型生命周期为核心的传统软件工程。Martin Fowler把传统软件工程方法称为重型方法，敏捷为轻型方法。可以合理推断敏捷宣言的意思是在传统软件工程已经得到了过程和工具的情况下，个体和互动更重要，并不是没有过程和工具，这点在 的《新方法学》一文中也得到了表述。如果不站在软件工程基础上来采纳敏捷，那么很容易沦为软件作坊，其实这是违背敏捷的。敏捷中给予软件开发团队最明确、最易操作的指导恰恰是在流程上，比如 P中的持续集成、TDD、重构等共12个实践，Scrum中的4种会议，FDD的5个流程等。分析这些流程，以及传统软件工程的流程，可以发现：敏捷把其中没价值的部分大大弱化了，有些甚至是取消了，比如中间文档里程碑评审、状态报告、需求矩阵等；而对其中有价值的部分，敏捷却是大大加强了，有些甚至鼓励追求极限，比如迭代开发、结对编程、代码规范、TDD等，而且有些敏捷流派还强调纪律。常常看到有些敏捷的材料中来比对经验主义的流程和已定义流程，说明在软件开发中，当过程过于复杂时，经验主义方式是适合的选择。但这里面存在混淆。已定义流程存在三种解释。

- 在Cockburn的《敏捷软件开发》第2版中，理论的或者已定义的过程是“一个理解得足够好可以自动化的过程（这个定义与CMMI对‘已定义的’一词的定义完全不同）。经验主义的过程需要人的检查和干预”。
- 在CMMI中，“已定义过程”的解释是“根据组织裁剪指南从组织标准过程集中裁剪得到的已管理过程，有得到维护的过程描述，为组织过程资产提供过程相关的经验”。
- 形成描述的过程是已定义流程。

显然，与经验主义流程比对的已定义流程采用的是第一种解释。在这种解释下，软件开发中的已定义过程是很少的，典型的有持续集成、每日集成、静态代码检查等，绝大多数过程需要人的检查和干预，无论是传统的需求分析、OOAD等，还是敏捷推荐的TDD、结对编程等。所以经验主义的过程并不是不要描述，需要而且必须要一定形式的描述。停留在个别团队成员头脑里的经验主义的流程是无法讨论传播的，而只要说出来，哪怕没有写下来，这个流程就已经得到了描述。“只可意会、不可言传”的东西是佛学考虑的问题，不是软件开发考虑的问题。不了解上下文的朋友会很自然采用第3种解释，那么极可能会误以为敏捷不需要过程描述。

因此这里真正的问题在于流程描述的颗粒度不同。口头表达的流程往往是颗粒度最大的，也是最不稳定的，而传统软件工程中包括进入—任

务步骤—确认—退出（ETV 范式）再加上详尽文档模板+评审+度量等的流程提供了很细的颗粒度，被Martin Fowler等称为“繁琐滞重”。细粒度的过程有更好的指导性，但显得繁琐呆板，粗粒度的过程往往是抓住关键，但对细节往往指导不够。按照刚刚好（Just enough）的原则，敏捷类流程描述倾向于刚刚好的粗颗粒度，整体上要讲究平衡。

得到描述的流程能够处理大多数情况，一

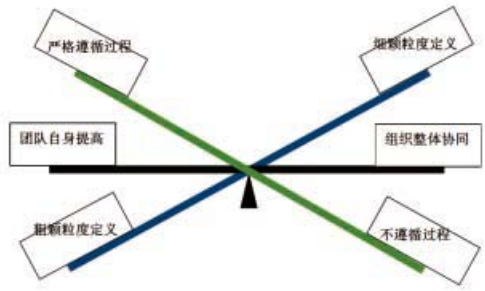


图3 过程的三组平衡

般的颗粒度粗的流程适用性更宽些，但无论如何是不可能把所有未来流程执行时碰到的情况都说明，所以就存在如何遵循流程的问题。显然在道理上，碰到新情况不能死硬地遵循流程，碰到老情况也不能肆意破坏流程。虽然恪守约束，避免整个流程体系走向混乱很好，但过于教条以至于无视实际情况同样不利于项目和团队。 根据精益的理念，我主张：“过程中发现浪费并积极消除重于遵守既定教条。”流程的严格程度要根据团队和团队碰到的实际情况来处理，不要忘记敏捷宣言的第一条价值观：“个体和互动**高于**流程和工具”。在流程方面还有一个问题，就是流程的定义和执行不完全是项目团队级的事情，就算是最尊重团队决策、已经采纳敏捷的组织也会倡导团队之间的经验分享，而有些组织就会直接要求团队采纳其他团队获得的流程，这些流程在不同组织有不同说法，比如有效实践、最佳实践、规范、规程、经验分享、知识共享等。这里我提议“组织整体协同并重于团队自身提高”。 把流程发挥最大作用，必须处理好如图3所示的三对平衡（说明：图3只是为形象说明存在三组平衡，高低位置并不代表哪个更重要，寻找合适平衡点是关键）。

从Scrum看，Scrum本身给出的过程略偏向于细颗粒度，比如站会、燃尽图等，而围绕着Scrum的不少培训材料和文章对计划会、反思、评审会等给了很多指导，感觉更加偏向于细颗粒度。在Scrum的咨询、培训和文章中，一般要求遵循Scrum给定的过程，对裁剪Scrum中的元素要慎重，尽量不要裁剪。Ken Schwaber对此的描述是“对Scrum规则的修改，只有在Scrum Master确信团队足够深入掌握了Scrum的运行原理，有足够的技能和思维来修改规则时才可以进行”。Sprint计划会上制定的计划通常被假定为某种承诺，作为承诺往往意味着务必要完成。在2011年7月发布的最新Scrum指南中，澄清那不是承诺，而是可以完成工作的预测，而且这个预测会在Sprint过程中因为更多信息而改变。我认为这个澄清非常好。承诺并不能准确预测未来发展，但却要求（甚至于是压迫）开发人员必须在限定时间内完成某些任务。澄清为预测后，相对容易改变，执行起来不再那么生硬。

从CMMI看，CMMI本身对各个过程域给出了目标、实践和子实践，看起来颗粒度上偏向于细，而且CMMI覆盖的过程范围比几个常见敏捷软件开发流派覆盖的范围要大，所以整体上显得也细。对待裁剪，CMMI在满足目标的情况下是允许并鼓励的，CMMI全部目标的累计字数与敏捷宣言+原则的字数在同一个数量级上，偏向于粗粒度的敏捷类过程描述也是能够符合CMMI要求的。

结束语

敏捷宣言和原则及主要的敏捷流派指出了优化的方向。根据实际情况进行适配，把握恰当的平衡，同时保持对优化方向的追求，保持在“ing”进行中，持续发现不足并提升，这就满足了敏捷软件开发宣言和原则。平衡、适配是本文强调的关键词。

张克强

系统分析师，Certified Scrum Master，硕士。现任宝信软件技术中心项目总监，参与了由中国敏捷开发联盟发布的《2010年国内外敏捷发展报告》的书写。

敏捷交互设计

文 / 熊子川

我们说说敏捷思想和实践在交互设计过程中的体现。

需求分析中的问题

采用文档传递需求，主要遇到需求失真频繁，实现难度太大，按照模块书写文档开发人员不知道全貌，端到端测试完等。解决方法是让需求人员和团队坐在一起，用用户故事和场景串联特性，频繁进行反馈，开发和分析人员结对写用户故事，频繁验收等。实际做了以下几件事情：

- 让他们对交付负责；
- 让他们一起工作；
- 让下游控制上游质量。

现在想想交互设计团队的问题。他们的东西只是一层皮，至于怎么实现他们根本不关注，但是客户对他们的东西很有感觉，把它们当做合同来验收。此外，他们的东西真的是用户想要的吗？

如果我们审视一下以上情况，会发现这里面提到以下几方面问题：

- 他们不对交付负责的风险；
- 用设计作为合同约定的风险；
- 设计本身质量缺乏验证的风险。

对交付负责的风险

交互设计本身具备两个特点：首先，人员间的技能区分明显，在国内对信息架构、交互模型、内容策略、用户研究和代码实现全部胜任的人才极少，人员可替代性非常低；其次，软件开发对交互设计的要求往往集中出现在前期，交付中的需求处于变化状态。

人员不可替换，需求量不稳定使得在很多情况下不可能将交互设计人员全部安排在一个交付团队里。于是往往交互设计人员来自于另外一个部门，他们会在项目开始后离开项目。

当他们只需要对上游，也就是客户负责时，实现的成本便不在他们的考虑范围，这本身就是一种高风险的承诺，最终的结果损害交付团队，究其最终结果损害的依然是客户。

在敏捷交互设计中有两个策略来解决这个风险，很多情况下，二者同时进行。

- 让对交付和运营直接负责的人拥有相当的话语权。软件的成功交付永远是用户价值，商业价值与技术成本和运营成本的博弈，必须让那些未来交付项目和运营产品的人全程参与到设计过程里来。

让更多的人参与到设计过程必定带来设计过程的变化。在一个典型的敏捷交互设计场景下，协作式的设计（Collabrative Design）是贯彻整个过程的核心，而单打独斗，加班赶工的设计方法将被摒弃。

- 让设计师在一定程度上对交付负责。这里的对交付负责分为两个层次：客户层次上，让主设计师成为客户代表，和客户建立起持久稳定的关系，管理好客户期待；交付层次上，定期回到团队，回顾之前产品设计方向，演进式进行调整，直接参与到交付中来。

这些都会交互设计团队提出新的要求。

首先，最大的变化是工作方式，躲在隔间里独自设计再交付文档模式将被改变，你的设计将全场面对所有人的挑战，你是否可以应对。

其次，尽量鼓励人员技能互换，培养T型人才，减少因为技能依赖造成的资源使用不充分。

最后，对交互设计团队主设计师提出更高要求。他不但要掌握所有交互设计相关技能，更需要相当的引导能力、讲演技巧、项目管理经验和处理客户关系的能力。

设计作为合同约定的风险

设计很容易被理所当然地认为是一种交付的约定——如果你告诉我未来上线成功后它是这样的，那我自然认为这是你们对未来交付的承诺。而风险就在于：

■ 客户看到的往往是表现，对于其中的细节，并没有被约定；那么真正让它工作起来，在设计阶段没有人知道成本会有多少，特别是当设计师完全依靠自己的想象，而无开发人员参与，更会造成成本变动的巨大风险。

■ 看起来过于完美的设计使得客户对于项目期待盲目乐观，同时使得客户的参与度降低。除了可以挑剔一些样式的问题，面对一个完美的设计方案时，客户很难在业务和体验上提出建议。在很多场景下，客户提出最多的是“看备选方案”和“先做出来再说”，如此这般，交付团队面临巨大的需求变动成本。

在敏捷交互设计方法中，我们所采取的策略是尽可能地让客户和交付团队代表参与到设计过程中来，采用“协作设计（Collabrative Design）”的方法，与客户一起发现和分析问题、设计和测试解决方案，最终达成一致。

换句话说，和客户在交付范围和细节上达成一致的最好方式就是和客户一起设计，即对交付范围进行确定、细节进行细化。

这里介绍一下“协作式设计”的概念。

“协作式设计”的核心观点是抛开已有功能需求列表的约束，和客户一起重新发现和分析问题。一切设计的过程不应该从解决方案开始，这是优秀设计产生的前提，如亨利·福特所说：“如果我问他们要什么，他们永远会说要一匹更快的马。”客户往往习惯于从讨论解决方案开始，这也是敏捷交互设计方法和传统设计方法最大的不同。

整体的过程大致可以分为四个阶段：分析、识别、设计、测试。

■ 分析阶段中首先抛开现有的功能需求，重新回到原点，从客户的典型消费者出发，了解典型消费者的驱动（可能促使他与所提供服务交互的驱使），目标（消费者期待从所提供服务中获得的价值），顾虑（阻止消费者与所提供进行交互的阻碍），通过这些，去还原消费者（图1）的典型体验。消费者体验将引导整个设计过程。

■ 在还原出消费者体验之后，我们会优先从两个方面进行识别：首先，那些现有消费者未能有服务进行覆盖的地方，简称机会；以及那些可以在消费者体验中改正的地方（图2），简称痛苦。换言之，设计的目标无非提供新的体验和提升现有体验两个方面。其次，通过观察现有消费者的体验模型，去识别真正有价值的产品方向，而非简单对照功能列表进行设计。

■ 寻找到真正的产品方向后，我们将选择最重要的某些体验目标进行设计，这个过程特别强调客户的参与，这样的参与包括两个方面的形式：首先是鼓励客户不断地对设计过程提出自己的建议，和传统设计方法不同，敏捷交互设计中特别重视低成本设计的重要性——尽量使用低成本的设计工具客户持续不断地展示成果，期待低成本的设计中间产物（图3）能够让设计过程更加透明，客户参与感更强；其次是给予客户直接机会对某个问题进行设计，我们会建立其不同的小组，由客户进行主导，充分发挥客户的创造力，选择最优方案进行细化。

■ 一旦我们产生了任何形式的设计，我们会对设计本身进行精细化，制作出原型，第一时间进行消费者使用测试。跟敏捷软件开发一样，敏捷交互设计强调尽早实现反馈机制，即在最短的时间内，推出具有端到端测试能力的原型，进行测试。这里的端到端便是由消费者体验驱动——不需要完整的所有交互细节，只需要在一定范围内实现完备的体验，便可进行测试。测试的结果将被加入到下一个测试迭代中去，迭代式的演进设计。

注意，这四个阶段中，客户都会全程参与。“协作式设计”保证了设计过程完全透明化，让交互设计的结果符合客户的期待，在交付范围上第一时间达成一致，同时设计本身是演进和变化的，它并不代表软件交付的合约。

设计的质量风险

回想一下敏捷软件交付中，如何进行质量的把控，敏捷交互设计也遵循类似的原理。首先我

图1 通过研究典型使用者设计体验

图2 在体验中寻找设计机会和改进点

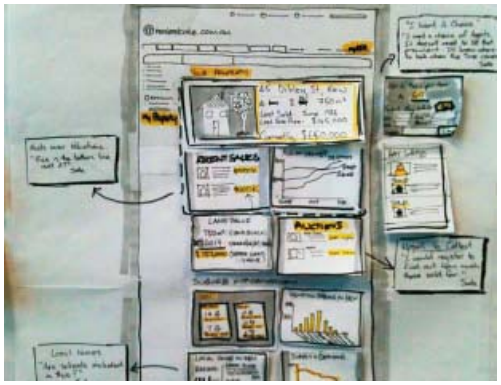


图3 低成本的手绘图进行交互设计

们看看敏捷软件交付中的做法：

- 持续集成的方式第一时间发现缺陷；
- 尽可能早的端到端测试；
- 尽可能小而频繁的提交；
- 全团队工作模式，避免沟通产生的失真；
- 尽可能短的反馈环；

再看敏捷交互设计是如何在类似的情景中实现对交互设计质量的把控。

首先，交互设计的结果自然无法进行自动化测试，那么此处的第一时间发现缺陷就是能够在最短的时间里发现设计中可能的缺陷。跟代码分层测试一样，交互设计本身也可以进行快速的分层测试：

内容层：可以采用“卡片排序（Card Sorting）”的方式和消费者一起对应用本身内容分组进行测试，什么样的内容分组更容易让用户接受；

信息架构层：可采用“页面流图（Screen Flow）”的方式用线框图还原出用户使用应用的体验（图4），收集反馈；此外“纸上原型（Paper Prototyping）”也是一种很好的方式测试不同信息架构对用户的影响；

交互层：可用HTML+CSS以及简单jQuery制作高保真原型，对交互特别复杂的体验进行具有交互性的实质操作性的测试；

视觉层：将视觉方案和高保真原型绑定在一起，尝试将视觉体验加载到真实交互测试中，尽可能模拟使用场景。

注意，这里的分层测试完全是迭代式演进的（图5），每次收到的反馈将会影响到下一轮的使用者测试中，而且测试精度也是逐步精化，目标是最短时间里实现全层次覆盖的高仿真度测试。

其次，测试过程必须要贴和使用者真实的端到端体验，原型的建造策略完全以最快完成一个用户期待体验为目标，后续的设计都围绕在如何丰富这个体验而进行。

当体验作为测试的脚本，更容易时测试者产生带入感——是否能从现有原型提供的体验中获得所期待的价值？如何更好。这与传统方式不同，传统方式更多是功能的组合堆砌，而没有明确的体验目标进行串联。

有端到端体验进行串联也使设计过程趋于收敛，更好地控制设计的范围，杜绝过度设计。特别是当交付计划本身也是与体验强相关时，我们的设计本身也是在设计交付，这将很好地控制客户的预期和达成对交付范围的共识。

再次，设计本身是增量式的演进，在敏捷交

互设计中，我们避免一个完整的解决方案，只在最基础的层面上做架构的设计，比如说内容组织策略、信息架构、和基本交互模式。而并不会面面俱到涉及所有功能。

图4 利用页面流图还原用户交互体验

每次提交进行测试的设计尽可能少，保证一个测试迭代为一周，演进式地进行设计。

同时“持续设计”的理念也是敏捷交互设计有别于传统方式的核心。在项目启动前，交互设计只覆盖核心体验的交互，而不对整个产品进行全方位细致的设计。真正当项目进入开发阶段，和开发并行的还应该有一条交互设计的工作流，负责在交付过程中持续对产品进行设计，对每个迭代交付的代码进行用户测试。当然真实场景下，交互设计的工作量不可能达到完全饱和，这里必然需要设计资源的有效利用。

图5 迭代式演进原型的精度

第四，正如之前所说的，敏捷交互设计团队必须是全团队工作模式，只有技能程度的区分，而无明确的职责分工。全团队的工作模式包含：

鼓励结对和轮换：鼓励有不同技能侧重点的成员进行结对工作；鼓励不同成员有机会接触自己陌生的工作；

所有产出物都由一个团队产出：避免第三方的资源要求，尽量保证这个设计团队产出完整的设计产出物；

活动由所有人参与：敏捷交互设计倡导互动性更强的工作坊模式，让所有团队成员加入到活动中去；

避免半成品的交接：尽量减少半成品的交接，保证一起完成某套设计的所有部分，避免半成品交接时造成的失真；

当然，建立其这样一支全团队，需要团队建设者更多的努力——如何让团队成员快速成长？如何让原有的“闷头苦干”工作模式向“合作开放”转变？如何营造出一个协作重于职责分工的团队气氛？这些都是团队骨干应该思考的问题。

最后，敏捷交付设计中提倡尽可能短的反馈环，第一时间发现改进点，这里面包含以下几点的实践。

尽可能可视化的工作环境：所有工作过程中的成果都应该在第一时间进行展示，这也是为什么我们推荐手绘或者线框图这样低成本的设计手段，它们可以更容易地进行可视化；

客户和用户的及早参与：通过协作式设计将客户加入到设计过程中，通过迭代式的用户测试，让反馈更加及时和有价值；

通过澄清或演示来讲述设计而非文档传递：尽可能多的使用澄清和演示的手段展示设计思路，少使用文档简单传递，传递设计细节最好的方式是一个强壮的原型；

这些，就是敏捷交互设计期望避免的三个风险，它们是：不对交付负责的风险、设计即合同约定的风险、设计质量无法保障的风险。

当然在每个环节当中必然涉及到很多具体的，且与传统模式不尽相同的实践方法，还包括这样一支团队应该如何建设等问题，在这里我并没有详细谈到，这里只是在概念的层次上向你介绍敏捷交互设计。有机会，还会讲讲各种实践的故事。

熊子川

ThoughtWorks高级咨询师，敏捷顾问，作为业务分析师参与多个敏捷交付项目，近年来一直致力于敏捷方法在国内的推广工作，现为一家国内知名的电信设备生产商提供敏捷咨询服务。

一地鸡毛

软件项目中的人际困局

文 / 方坤

作者结合切身经历，展示了他之前所在团队软件项目延期的种种原因，而其中印象最深刻的是各种人事纷扰乃至于勾心斗角。

六年前，毕业未久的我在一家外企工作，我所在团队开发的软件项目在交付到集成测试组时因种种原因延期一周。这本身根本不是什么大事情，但其间各种人事纷扰乃至于勾心斗角却着实令我印象深刻。

公司

我的老东家是一家大型跨国电信设备开发商，曾具有辉煌的历史。我还记得在公司110周岁的生日庆典上，一位高管致辞说：“110年，这不是奇迹，是成绩”，令人不胜欷歔。遗憾的是，公司在.com泡沫中遭遇重创，一蹶不振。时任CEO为求摆脱困境，打起了人力成本的主意。当时，公司在美国雇用一名工程师的综合人力成本接近中国的2.5倍【注：工资只是其中一小部分】。至于法国，成本比美国还要略高一些，而且不要忘了，人家可是35小时工作周。大家都是聪明人，很快就看到端详：公司正在法国裁员，将项目转移到中国。

令人尴尬的是，我所在的中国团队恰好就在与法国团队合作。这一项目最早完全在法国，此后几年时间，中国团队大规模扩张人手（我就是这样进来的），将项目模块逐一从法国团队手中接过来。刚开始，法国工程师将原先的模块移交中国之后，便转而从事其他项目或职位，谈不上什么个人损失，双方共事可谓融洽。后来可就不是这么回事了。有一次，两位中国工程师去巴黎接手一个项目，一位法国

工程师负责培训，为时2~3个月。在这位法国老师兢兢业业的帮助下，两位中国工程师成功掌握整个模块，按期于圣诞节前夕归国。告别巴黎时，没有一个法国同事去跟他们寒暄话别——那位法国老师被裁掉了，他的最后一个工作日恰好就在两位中国工程师离开的同一天，法国同事都去送他了。

到发生本文将要详述的交付延期之时，所有模块的开发工作都已从法国团队移交到中国团队，而集成测试虽然仍由法国团队负责，但从法国到中国的移交也已开始。不妨猜猜看，法国集成测试团队的工程师们此刻在想些什么。

团队

我很幸运，毕业后初入职场就遇到一位好经理H，坦率地说，她也是我到目前为止跟随过的几位经理中最好的一位。中国很多女经理都有一个共同的特点：没有私心。她们对于自己的晋升、提薪并无多大热情，更愿意把心思、时间和精力花在辅导和培养自己的团队上面。

H因分娩而“暂时”离开我们团队。经过短暂的过渡，接任我们经理的是T，一位新近招聘的职业经理人。他的风格与H大为不同。仅举一例说明。当年向H请一天年假，她总是微笑着说：“没问题。不影响工作吧？”T则会端起架子：“不影响工作吧？没问题。”语序上的变化，加上语气的差别，虽然只是细微末节，却反映了态度的不同、对员工是否尊重。除此之外，更严重的是工作态度问题。现在我们知道，T在北京待了不到一年时间，买下两套房、一辆车，还办妥了到加拿大的移民和那边的工作——而在当时，我们这些员工仅仅只是知道，我们的经理不太在办公室出现。

在团队内部，我所在的FM小组与另一个CM小组工作是紧密衔接的。但在CM小组的核心员工之间却存有罅隙：小组长B与技术骨干S矛盾日增。怎么说呢，这两位都是很好的同事，然而好人之间也会彼此鄙视的：S认为B不懂技术，瞎指挥；B认为S目空一切，难以共事。缺少一位好领导来调和，好员工也不能组成一个好团队。

流程

我们开发的是一个庞大的电信软件项目——3G接入网网管系统，采用的开发流程仍然是传统的瀑布式。简单来说，依时间顺序，一个软件工程师（首先是各小组的小组长）需要依次参与以下几个阶段。

- **需求阶段**：跟踪和审阅由系统架构师撰写的需求文档，必要时要求澄清，然后预估工作量，经理据此调整人员安排。
- **设计阶段**：分析需求文档，完成模块设计，据此撰写高层设计文档和底层设计文档，前者以定义模块接口为主，后者则涉及更多细节。
- **编码阶段**：根据两份设计文档完成实际编码工作。
- **单元测试阶段**：是的，你没有看错。根据本部门正式的、成文的流程，单元测试阶段在编码阶段之后安排时间进行。在实践中倒是没有这么僵化，大家尽可以测试先行，只要时间大致齐即可。

各开发团队在完成各自负责的一或多个模块的单元测试之后，将代码提交到统一的代码库，打上标签，然后将这些标签连同其他注意事项写成文档保存到指定目录。其后，就是集成测试阶段了——集成测试团队收集所有团队的所有标签，从代码库提出相应的代码进行编译，编译成功后即按照事先准备的测试用例进行测试，给开发团队提Bug。

我参与了前面几个版本3.X、4.0的开发，仅从技术角度而言，瀑布式开发流程工作得尚称流畅。但工程师是要领工资的，软件写出来是要卖钱的，一套经典的瀑布式流程走下来往往耗时几个月甚至年余，等到软件产品正式发布，用户需求已然发生变化，这怎么赶得上趟呢。公司不是没有意识到这一问题，但舍不得做伤筋动骨的巨变，只愿意在现有流程上做一些微调，效果甚微。

有一个例子很能说明问题。当时，中国的销售部门向总部反映，我们在中国市场遭遇到本土厂商的强力阻击，要想争夺中国市场，就必须在定制化方面下更大工夫。在大中华区乃至总部高层的大力支持下，我们部门成立了一个“快速特性”开发小组，专门根据中国客户的需求为我们的产品添加相应的特性。有一个快速特性是这样的：本来，我们的网管系统会在电脑屏幕上显示一台虚拟的机器，如果某个部件坏了，代表该部件的绿灯就会变成红灯并开始闪烁，提醒操作员注意。中国客户看过演示后说不错，但光红灯闪烁还不够，还应该放点儿警报声出来，不然操作员离开座位了怎么办。我们的销售一口答应下来。猜猜这个特性我们做了多久才交付给客户？三个月！这就是瀑布式流程下的“快速特性”！（当然，中国的销售部门和开发部门分别向国外的上司汇报，由老外负责协调中国的事情，这也是造成拖延的一个同等重要的原因。）

这样拖拖沓沓做出来的产品，其销路如何不问可知。公司应对的办法，就是一方面推新版本、新特性来吸引客户，另一方面强调开发速度的重要。很显然，这两者之间存在矛盾：新特性越多，开发时间就越长，客户不会买账；可是如果新特性太少，跟上一版本差异不大，客户同样不会买账。

版本

有一天，经理通知大家：咱们要开始做新版本4.1了。其后，像往常一样，我们就陆续接到一批批需求文档，开始预估工作量。然而这一次，事情一开始就有些不同：这些需求文档写得异常混乱，常常不知所云。我们如何能够根据一份看不懂的需求文档来预估工作量呢？我们随后联系法国的系统架构师（他们处境安全，跟我们合作融洽）要求澄清，他们也很不好意思，有时还确实能够做出澄清，但大多数时候要么含含糊糊地来一句“我们也在研究”，要么说“根据我的经验，这条需求应该与你们模块关系不大”云云。大家就这么半猜半蒙，在磕磕绊绊中前行，心中满是不详的预感。

4.1版本很仓促地做出来了。又有一天，经理通知大家：4.1版本过于保守，连公司自己都觉得销路不会好，决定立刻开始4.2版本计划。于是一切都重新来过，而这一次，情况更糟：大多数需求文档都是匆匆写就，语焉不详。一些需求文档只有一个标题，正文人家根本就没来得及写，而经理就要求我们根据这样的需求文档来预估工作量。如果你认为这样很夸张的话，那我只能批评你想象力有限——个别文档只有一个编号，收纳到某一领域的写作计划之中，连标题都没有，而我们仍然要据此预估工作量！就这样，日复一日，我上报一些连我自己也不相信的数字给经理，而他则努力装出相信的样子。

软件工程应该怎样做？我本来以为CMM、TL 9000等是王道，经历这一风波我才深切地体会到，再好的流程与制度也经不住扯淡啊。人和最重要。

4.2版本更仓促地做出来了。又又有一天，经理通知大家：4.2版本过于激进，工期又短，从设计到实现毛病多多，公司也不看好，决定立刻开始5.0版本计划。这一次倒是没有太荒唐的事情发生。5.0版本实际上没有什么全新的特性，而是将4.1、4.2这两个版本的特性做一折中，从这个意义上讲，叫它4.1.5版本更合适，当然这个话不能对客户说。就这样，几个月以来第一次，大家终于能够做点儿靠谱的事情。然后，出事了。

风起

第一波的事故，我是直接责任人之一：因为我的失误，我负责的FM模块没有通过编译。

我还记得前几个版本交付时和H一起工作的情景。她会敦促我们尽量提前完成开发和

测试工作，提交代码，打上标签，撰写交付文档。她会亲自检查我们的交付文档，连一个细节也不放过。比如有一次，她就发现我无意中开启了Microsoft Word的中文自动纠错功能，把“...”（在版本配置中具有特殊含义）自动替换成了半个中文省略号“…”，让我脸上无光。大大小小的问题被她连续抓住几次之后，我开始小心谨慎，此后几个版本都顺利过关。印象最深的还是编译时的一次次待命。由于时差的关系，法国同事依据标签提出代码开始编译的时间是在北京的晚上，每一次，H都会带领我们几个少数技术骨干在办公室待到夜里，直到我们团队负责的所有模块都成功编译之后才离开。这可真是一件苦差事，而且在我当时看来毫无必要——我们的模块从来都是一次编译成功，错误（如果有的话）从来都属于其他团队。有一次，法国同事连续犯错，导致编译迟迟不能开始。当时我还保留着自校园带出来的早睡的习惯，时间一长，上下眼皮开始打架。H就跑到我的座位，谈人生，谈理想，谈八卦，反正就是不让我睡着。一直坚持到凌晨两点，编译开始之后照例一次成功，H才领着饥肠辘辘的我们离开办公室，请我们到楼下的小店吃宵夜。喝着温暖的豆浆，我在心中嘀咕：“真是事儿妈啊。”

这一次，“事儿妈”不在了，新任经理给予我们“完全的信任”，从头至尾都放开手——这同一件事情我们都连续做了好几遍了，还能出什么错呢?

还真就出事了。前几次，我们至少能够提前一周左右的时间完成全部工作，这抢下的一周时间足够我们反复测试、排查问题，并为应对突发事件留下时间——尽管突发事件从未发生。而这一次，大家经过连续几次折腾之后疲惫不堪，工作效率低下，更何况这次的工期本来就偏紧，还被前面几个环节挤占不少。我们FM小组勉强提前几天完成工作，CM小组却陷入苦战，加班加点，紧赶慢赶才在最后一天完成。FM模块依赖于CM模块，这样一来，我们也受到连累，不得不换上CM小组最新的标签，重新测试FM模块、打标签、修改交付文档。等到我饿着肚子敲完最后一个字符，又仔仔细细检查了几遍，已是周五晚上7~8点钟的光景。我长吁一口气，站起身来，摇摇晃晃地离开了办公室。

等到我周一早晨回到办公室，这才发现自己犯下低级错误：我忘记将修改后的交付文档保存在指定目录了！这样一来，法国同事据以编译的乃是先前保存的老版本的交付文档，FM模块编译失败！我赶紧寄出道歉信，连同最新的交付文档。然而，晚上的编译仍然没有成功。根据法国同事提供的编译错误日志，我很快就发现问题：FM模块与其他依赖模块之间使用了不一致的标签。说起来还是怪我们两边当时掉以轻心，只是口头约定了一下，也不知怎么就听岔了，关键时刻害人。又是一番折腾，FM模块在第三次编译中顺利通过，我心中一块石头才算是落了地。

乱战

我这边没事了，CM小组却开始焦头烂额。

CM模块几次编译均告失败，而法国同事提供的编译错误日志乱七八糟，毫无帮助。原来，我们项目当时尚未采用分布式编译技术，为了缩短编译时间（仅仅某一个子模块单机重新编译就需要18小时），法国的集成测试团队自行编写了一个脚本，开启几路进程并行编译各个子目录。这个脚本写得过于简单，几路进程的输出信息全都杂七杂八搅到了一块儿，以至于CM小组研究了几天，连到底哪个子目录编译不过都没闹明白!

CM小组尝试向风雨飘摇中的法国集成测试团队请求帮助：“你们能否用单路进程编译CM模块的各个子目录，将错误信息提供给我们？”

法国人回答：“请中国团队尽快修复编译，你们堵住了整个项目！”

CM小组解释说：“我们正在努力，你们能不能帮忙……”

法国人回答：“请中国团队尽快修复编译，你们堵住了整个项目！”

CM小组再次尝试：“这一错误本地不能复现，而编译日志……”

法国人回答，并且抄送各路神仙：“请中国团队尽快修复编译，你们堵住了整个项目！”

外事不靖，内部也不安宁：CM小组的小组长B和技术骨干S此刻正在斗气！从一开始，B就将编译错误的排查工作分配给自己和另一位同事，没有邀请S介入，而S也不主动过问。没想到这么一个乍看上去再简单不过的错误一拖就是好几天，这样一来，双方陷入僵局。站在B的角度，如果连个编译问题自己都解决不了，还得请S来当救兵，这不是坐实了自己不懂技术的指控吗，这张脸以后还怎么搁？再说S一直面无表情地坐在自己的电脑前做自己那一摊事情，一句问话没有，这不摆明了是要袖手旁观吗？而S也有自己的苦衷：自己要是一开始就主动介入倒也罢了，如果拖到现在才出手，那怎么解释自己前几天不闻不问的态度？就算自己辩解说确实没有端架子、看领导笑话的意思，完完全全是在服从领导安排，也得有人信啊！双方有一点想法倒是共同的：这个编译错误赶紧消失了吧……

既然CM小组迟迟不能修复编译，顺理成章地，项目经理（一个不偏不倚的法籍华人）开始找他们的上级，也就是我们共同的经理T。然而——她找不到T！事情就是这么凑巧，虽然T平时就神龙见首不见尾，可像这次这样整个礼拜办公室都不怎么见人影、写信也不太回的情况还真不多。连续几天，项目经理从法国给T的座机打电话，按说这是法国的休息时间，中国的上班时间，可是法国那边有人打，中国这边没人接。电话留言、电子邮件都不好使。项目经理急了，电子邮件写得越来越不客气，每封信的结尾都是同一句话——“T在哪里！”……

事情终于惊动了上面，领导出来问话了：“发生了什么事？为什么会耽误到现在？”法国团队再次暗示中国团队无能，中国团队则强调本地无法复现，必须法国团队配合，项目经理在居中调解的同时狠狠地告了T一状……领导不愧是领导，跳过T的事情不提，和蔼可亲地建议法国团队考虑中国团队的合理要求……事情终于走上正轨。法国团队终于按照CM小组的建议尝试单路编译；与此同时，B主动去征求S的意见，问他是否愿意参与排查，而S也立刻答应下来；T又神秘地出现在办公室里，如果这有关系的话……经过整整一周的纷扰，周五，编译终于成功。

那么，这一编译错误到底是如何产生的呢？说起来，这居然还与前述混乱的版本计划有关。在4.0版本中，出于兼容旧有设备的需要，CM模块中有些文件按照foo_V4.h的格式命名，后来升级到4.1、4.2版本后文件内容相应修改，文件名保持不变。可是5.0版本实际上是4.1、4.2版本的综合，CM小组被迫把4.1、4.2这两个版本的foo_V4.h文件都引入5.0版本，文件名分别命名为foo_V41.h和foo_V42.h以示区别。换言之，文件名变长了一个字符，而这就导致法国集成测试团队的编译脚本中的命令行超过了最大长度的限制……

尾声

软件工程应该怎样做？我本来以为CMM、TL 9000等是王道，经历这一风波我才深切地体会到，再好的流程与制度也经不住扯淡啊。人和最重要。

无论版本号如何，我们的产品终究还是销路不畅。新任CEO上台后，大刀阔斧厉行改革，将整条产品线出售。毕竟，对于IT业来说，创新才是利润之源，单纯的削减成本没有出路。基于这一认识，我转投互联网公司，从此踏上新的征程……

方坤

Google中国资深软件工程师，清华大学电子工程系硕士。曾先后在创业公司与大型外企任职，2006年加入Google中国，先后从事过基础架构、产品搜索、音乐等项目的研发和运营维护。

责任编辑：董世晓（dongsx@csdn.net）

Marty Cagan
过去20年，Marty Cagan作为负责定义和开发产品的高级经理人为多家一流企业工作过，包括惠普、网景通信、美国在线、eBay。他亲历了个人电脑、互联网、电子商务的起落沉浮，致力于通过写作、演讲、培训帮助客户打造富有创意的产品。为此，他撰写了《启示录》一书，创办了硅谷产品集团公司（SVPG）。在此之前，他的最后一份工作是担任eBay产品管理及设计高级副总裁，负责规划全球电子商务网站的产品和服务。

做个优秀的产品经理

文 / Marty Cagan 译 / 胡倩，潘希颖，周蓉

Marty Cagan是享有世界声誉的产品管理专家，曾经担任网景副总裁、eBay产品管理及设计高级副总裁。本文是他回顾自己二十多年来从事软件产品管理工作的总结和经验分享，主题为产品团队的组建、人才的选择和评估。

建设公司从建设团队开始

我一直提倡设立严格的产品经理入职标准，因为他们的工作决定了产品和生意的成败。但我常常听到管理者抱怨公司的产品经理是以前的产品营销人员，这些所谓的产品经理具有我以前文章里提到的所有问题。要改变这种现状着实让管理者头痛。因此，我想谈谈产品经理管理者的角色和职责。

管理产品经理的人通常被冠以产品总监或产品副总裁的头衔，这是高科技公司里最重要的职位之一。产品总监对公司业绩的影响绝非他人可比。成功的产品可以开启新的业务方向，失败的产品则可能拖累公司破产。这份工作不成功则成仁，鲜有居于二者之间的。

产品总监的关键职责有两方面。第一，组建优秀的产品经理团队。第二，规划公司的全局产品战略，对产品组合负责。下面我分别讨论这两种职责。

建设产品管理团队

因为产品管理职位非常重要，所以训练和培养产品管理团队的工作能力不但是产品经理的任务，更是产品总监的任务。不称职的产品经理浪费开发时间、怠慢用户、辜负客户的信任，注定是要失败的。其他岗位可以容许不称职的员工侥幸过关，总能找到替补收拾烂摊子，但是大部分产品只有一个产品经理，想找替补都难。如果产品经理不称职，只能退而求其次，请其他团队成员（比如主程序员）越俎代庖。

如果你发现手下的产品经理总是无法胜任工作，就要立刻采取行动。有些人永远不可能成为称职的产品经理——他们就是不适合这个职位，无论如何培训、指导都无济于事，但对那些有潜力的产品经理，应该多花时间帮助他们提升管理产品的技能。不管怎样，产品总监要确保每个团队成员都没有掉队。

我认为新产品经理必须经过约三个月刻苦学习才能开始管理产品。这段时间，他要融入目标用户和客户群，学习相关的技术，了解市场和竞争局势。管理者应该为新人创造学习条件，监督学习进度。注意，这三个月还不包括对产品经理的技能和职责的培训时间。三个月的学习期也适用于有经验的产品经理，因为他们也需要熟悉产品特有的客户和领域。

设计一套培训计划，让新入职的产品经理充分接触用户和技术。已经入职一段时间的产品经理如果在这方面掉了队，也要让他们在工作之余补补课。确保他们明白自己的不足之处。

如果某位产品经理无法胜任工作，管理

者应该重新寻找合适的人选。解雇别人并不容易，但这是你的职责所在。为了团队、公司、客户的利益，你必须纠正现状，帮助下岗者找到擅长的岗位，同时擦亮双眼寻找新人接替他的工作。

一旦找到有潜力、称职的人选，就应该放手让他们工作，尽情发挥其潜力。如果你事无巨细都过问，他们就不可能步入正轨成为产品的主人。当然，一定的监督和帮助还是有必要的。如果你不相信你的产品经理，那就换一个可以信任的人。只要你给他们足够的空间，我保证你不会失望。

请注意，你必须确信产品经理有足够的能力，才能够授权给他们。授权给不称职的人，那是推卸你作为产品总监的责任；如果你事必躬亲，那是替他们承担责任。

聪明的产品总监知道，团队成员的出色表现就是自己的出色表现，所以要雇用比自己聪明的人，尽可能为他们创造宽松的工作条件。

规划公司的产品战略

产品总监负责管理公司的系列产品，决定公司经营什么产品，仔细评审每款产品的产品战略和研发流程。

他必须透彻理解公司最新的商业战略，确保产品战略直接支持商业战略；与产品经理一道完成产品规划，共同实现规划；带领产品团队建立产品原则，坚持按产品原则研发产品。

有了最好的产品经理团队，即使每个人的工作都非常出色，产品总监还是可能遇到产品间的冲突，毕竟每个产品经理只想优化自己的产品。产品总监必须设法识别、解决这种内部冲突。

产品总监要负责制订产品组合路线图——兼顾用户需求和商业目标，从全局出发制订产品发布计划。

最后，产品总监要处理好与公司同事的关系，特别是得到公司高管（尤其是CEO）的信任。产品总监是公司的关键人物，必须礼贤下士，集思广益，决策有理有据、公开透明。受人尊重的产品总监才能得心应手地解决意见冲突，抵制错误决定。

这是一项极具挑战的工作。工作在这个岗位上的人无一例外是公司最优秀的员工。

产品领导者的必备特质

我一直坚信这样一句话：员工因公司而入职，因上司而离开。这句话同样适用于产品经理。上司是影响员工工作满意度的主要因素。好的管理者要能提高员工工作效率，调动员工积极性。

在上一篇的文章中，我列出了优秀的产品经理应该具备的特质，这一次我则要强调一下优秀的产品领导者应该具备的特质，作为产品经理，产品副总裁以及营销副总裁的考核标准。事实上，任何高科技公司的高管都需要具备这样的特质。

> 产品总监的关键职责有两方面。第一，组建优秀的产品经理团队。第二，规划公司的全局产品战略，对产品组合负责。

对于领导者而言，这份特质清单可以作为自我评估的标准；对于那些立志成为领导者的员工而言，这份清单便是你努力的方向。

以下是优秀的产品领导者应有的特质。

- 致力于打造优秀的产品，设计良好的用户体验。
- 始终保持积极的态度，遇事淡定、永不放弃。
- 营造良好的氛围，让员工享受工作、乐于创新。
- 认真倾听，谨慎说话。
- 员工犯错时，私下给予批评，明确指出错误；员工取得成绩时，公开给予嘉奖，不要吝啬赞美之词。
- 言行合一、值得信赖。
- 保证上司的“知情权”，即使项目进展不容乐观。
- 做重大决策时先征求团队意见。
- 事前请示上级，不要“先斩后奏”。

■ 和上级讨论问题时，保持审慎的态度，给出建设性意见。

■ 遇到争议时，坦率地表达自己的观点。一旦做出决定，则全力以赴去完成。

■ 做个好伯乐，组建优秀的团队，对每位员工的职业生涯负责。

■ 工作出错时，迅速承担责任；工作取得进展时，归功于他人。

■ 努力自我完善——错误发生时，研究其他改善方法。

■ 不八卦上司，也不责难同事——尊重每一位员工。

■ 永远视客户为“上帝”。

■ 短期战术和长期战略，“两手抓，两手都要硬”。

■ 言行举止间传递企业价值观。

■ 任何决策都要以公司长远利益为目标。

■ 为建设伟大的公司而奋斗“终身”。

我必须承认，上述清单我自己也无法完全做到。但我常会以此自省，以便成为一个更好的管理者。即使是那些杰出的产品领导者，也不可能时刻保持完美，但他们总会努力地自我完善，向目标迈进。

以能耐论英雄

从本质上讲，产品就是创意，产品经理的职责是想出好点子并加以实现。这需要技巧和实践，难以言传。我们需要好点子，有些想法是我们自己的创意，但如果仅依靠自己，就会严重限制创意的发挥。

我有一点工作体会，做产品要找公司最聪明的人合作。我发现每个公司都有几个聪明绝顶的人，这些人是公司的潜在资源，关键看你能不能发现他们。如果有幸能找到他们，就应该不拘一格地任用。我把这些人看作产品副经理，甚至公开授予他们头衔，把他们招进产品团队。

为了说明公司的角落里隐藏着高人，我给大家举几个有代表性的例子，都是真人真事，只不过用了化名。

山姆 我花了很长时间才注意到他，因为他的经理一直对他评价不高。幸好很快就真相大白了——山姆卓越的才华使得他愚笨的经理感到地位岌岌可危。现在他的经理已经不知去向，而山姆则成为优秀的产品经理。

克里斯 我是在和同事一道拜访客户时遇见克里斯的。销售人员在介绍当地的情况时毫无章法，我们听得云里雾里，不知所云。后来，一位名叫克里斯的系统工程师（他负责为销售人员提供技术支持）“临危受命”，清晰、明了地阐明了当地的情况，令客户佩服不已。之后，我邀请他去喝一杯。杯酒之间我就坚信，对面坐的是个天才。我邀请克里斯为公司提供产品建议和创意。现在他是一家“财富五百强”公司的产品总经理。工程师通常对技术相当熟悉，所以他们往往对客户需求有极强的洞察力。克里斯不仅能深入了解客户需求，还能提出可行的解决方案。

艾里克斯 在开发团队中，艾里克斯显得相当低调。他害羞内向，也没什么野心，但聪明过人，重视用户体验。大家没有发现艾里克斯在产品创意方面同样才华横溢。在我的帮助下，他很快成为公司产品理念的先锋。

马特 即使是高新技术企业，也存在不同形式的歧视。年龄歧视是最不应该的。马特十几岁就大学毕业，并一直奋发向上。但我见到马特时，他并未被重用，因为他的经理认为这么年轻的小伙子无法承担重任。后来，马特跳槽与人合伙开了一家新公司，他们的产品改善了上百万人的生活。

米拉 她是个天才，却被认为有两个“缺陷”——女人、印第安人。在这个男性主导、技术驱动的行业里，女性常被轻视。而且大家认为印第安人性格温和，魄力不足。但米拉很快就从枷锁中解脱出来，成为一名出色的产品主管。我在华人当中也见过类似的情况。不要让文化差异或口音蒙蔽了你。

产品经理还可以向自己的领导借力，听取他们对产品的建议，虽然他们不太可能参与具体工作，但并不表示他们会袖手旁观。

你需要的帮手可能隐身于公司各处——开发部门、销售部门、客户服务部门，甚至董事会。如何发现他们呢？

■ 打听！多问问同事，肯定会有收获。

■ 采用走动式管理模式。这源于惠普的做法。管理者要走出自己的办公室和圈子，花时间与员工相处。

■ 认真倾听与会者的对话与发言。

■ 敞开办公室的门，让大家知道你随时欢迎他们向你提出产品建议。

■ 坦率地把你的烦恼告诉同事，大家会热情地帮助你。

■ 一起泡吧。工作之余，产品经理总是与产品经理一起消遣，高管总是与高管为伍，这是司空见惯的事。如果你能抽出时间与普通员工一起休息、娱乐，一定能发现“埋在沙里的金子”。

许多产品经理不愿接受这些建议，主要是因为过于自负。他们认为自己才是提出创意的人，如果由别人提出创意，还要产品经理做什么？虽然我也有些好创意，但更多的灵感是受别人启发得到的。记住，公司的目标是打造卓越的产品，所有可以借用的力量都是可取的。

怎样评估产品经理的工作？

常常有人问我如何评估产品经理的工作业绩。我相信唯一正确的评价标准是看产品本身是否成功。我知道这个答案难以让人满意，因为评估产品表现的方式难以确定。是按收益计算，还是按利润计算？是按用户数量计算，还是按页面访问量计算？这些指标从不同方面反映了产品的情况，但无法全面反映产品经理的业绩。

最近出现了一种考察业绩的新指标：用户净推荐值（net promoter score，NPS）。这个标准很简单，请访问http://www.netpromoter.com阅读相关信息。

它的原理如下。调查用户是否愿意向他人推荐你的产品，满分是10分。选择9~10分的客户称为推荐者（他们会告诉朋友非常喜欢产品，相当于在为你做宣传推广）；选择7~8分的是中立分子；选择0~6分的称为贬损者，他们不但不推荐产品，反而会在朋友面前诋毁产品。计算出推荐者所占的比例，再减掉贬损者的比例，就得到了NPS，它反映用户对产品的态度。

很多公司采用了这个指标，有些公司的NPS得分很高，如苹果、亚马逊、谷歌、eBay——这是意料之中的。

这个指标反映了产品的用户体验水平。当然，理论上即便拥有100%的满意用户，公司也可能因为在每个用户身上都亏损而破产。但是就单个指标而言，我认为它有利于让公司关注用户满意度。而且口碑营销是最有效、成本最低的营销方式。

这个标准还能用来区分优质收益和劣质收益。例如，赞助商或广告合伙人希望利用你的网站向你的用户群宣传他们的产品，这件事可好可坏，取决于如何达成。如果做得不好，短期收益可能很可观，但是影响了用户体验（NPS会体现出来），从长远看，业务增长会减缓。反过来，如果做得好（与广告合伙人密切合作），就会提升用户体验，会让业务增长更快。

这就是为什么特例产品是危险的。客户为了低价购买产品就必须承诺接受限制条件。特例产品代表了劣质收益，它会降低用户的满意度。

你可以跨公司甚至跨行业比较NPS，这是件有趣的事。不过NPS最主要的作用还是用来观察公司产品和服务的优化情况。如果你还没用过NPS，不妨一试，观察产品变化如何影响NPS。你需要慎重决策，考虑每件事对NPS的影响，提高用户满意度。

本文节选自华中科技大学出版社《启示录：打造用户喜爱的产品》一书和作者的博客。该书从人员、流程、产品三个角度介绍了现代软件（互联网）产品管理的实践经验和理念。特此感谢华中科技大学出版社与Marty Cagan先生授权。

如何使员工更敬业

刘伟
万兴软件副总裁

敬业，功夫在诗外

敬业有两层含义。一是个人层面，头悬梁、锥刺股，王国维读书三境界之“为伊消得人憔悴”，都是个人层面的敬业。二是组织层面，即营造一种敬业的组织氛围。一个人原来并不“好好学习”，到了你这里，很自然地就“天天向上”了，这就是组织层面的敬业，“随风潜入夜，润物细无声”。

公司不同的发展阶段，对敬业的要求不同。初创阶段，员工比较少，创始人冲在一线，公司出不起太多钱请精英，选人原则是“敬业第一，能力第二”，待遇甚至达不到行业平均水平，但大家愿意干，在方向正确的前提下，一般就能壮大。发展阶段，队伍扩大了，创始人在后面激励大家干，公司有实力请得起高手，选人原则演变为“能力第一，敬业第二”，但前提是敬业能够上升到组织层面，形成勤奋、努力的组织氛围和土壤，于是即使吸引大批人才加盟也不会冲淡原来的创业文化，继续保持激情和活力。

我们团队一直保持着旺盛的战斗力，勤奋努力的敬业精神常为投资者称道。能做到这些，仅靠高层的榜样作用是不够的，公司层面的敬业问题，还是要从公司层面解决。

价值共享是基础。我们推崇价值共享的分配原则。一方面是共享事业发展的机会，用人机制灵活，大胆启用新人，通过项目工作给予表现机会，赛马不相马，做出成绩，即充分授权给予重用，由此赛出不少“黑马”；另一方面是共享公司发展的物质成果，实施员工持股计划。通过这种价值共享机制，使得大家的“敬业”有了动力基础。

文化建设是根本。我们推行平等参与的“村落文化”，基本做到“我的地盘我做主”。公司的重大决策，特别是涉及员工重大利益的事宜，员工均会参与乃至主导决策。我们的年度战略讨论，就有数十人进行了两天的封闭式讨论，为达成战略共识奠定了共同的前提假设；员工持股计划，由15名核心人员组成员工持股委员会，组织公司大多数员工积极参与，经过数十场次激烈讨论，才形成员工持股名单。

干部队伍是关键。员工是否敬业，一线管理人员营造的微观管理氛围很重要。我们有专门实施干部管理的部门，每年进行一次干部上岗评审，下属评价占40%的比重。这对干部的管理能力提出了很高的要求，要让员工跟着你感到有奔头。如果得不到员工信任，很快就会“被转岗”。公司每年进行盖洛普Q12工作环境测评，根据测评结果，帮助干部改善管理技能。如果说干部上岗评审是结果管理，不能获取大家信任就得下台，那么Q12测评与改善就是过程管理，要有改善计划来持续提升工作环境的员工满意度。

价值共享机制，平等参与的文化，干部良好的管理技能，做好这三点，员工敬业度就是水到渠成。

敬业必须要以身作则

我们是做人才管理软件的，因此对人才非常重视。我们的技术团队在招聘员工时有一个很明确的要求——把人的素质而非技术放在第一位。谈到素质，首当其冲的便是员工的敬业精神。

在我看来，敬业更多的是一种激情。如

张庆化
北森CTO

果仅仅满足于那种朝九晚五的工作模式，工作是工作，生活是生活，分得很开，这样的人可能就不适合做技术，因为技术工作更多的是需要对技术有特别的激情。举例来说，我们一位工程师有天晚上大概到12点的时候，打电话告诉我，他终于想出了某个系统的设计方案，要和我聊一聊。我很开心：这种工程师，能够把他的工作当成自己的事业或者乐趣，是任何一家公司所追逐的。

敬业必须要以身作则，不管是工程师还是主管，在工作中都必须做到这一点。在我们公司的技术部门，没有纯粹的管理者，每个人都必须写代码，只是根据工作职责不同，比例大小有区分而已。我们的管理者更像是一名教练，背后支撑着整个公司技术体系的发展。

而为了使员工更敬业，我们也会从两个方面进行引导。

一方面，按照员工不同的专长和领域，组织完善的成长阶梯，比如业务专家、技术专家等，让员工能时刻感受到自己努力的方向。对于有志于向某个方向发展的员工，我们会通过培训和导师制度帮助他成长。

另一方面，打造自主的工作环境，让工程师在自主的环境中自由发挥，让团队实现自管理。当自管理团队形成的时候，又反过来使工程师体会到更多的自主，从而对其从事的工作更加爱不释手。当然，自主是建立在信任基础上的，如果工程师犯了错误，我们会帮助他们总结经验，使他们走得更远。给他不同的挑战，通过不同的方式帮助他成长，更有利于他爱岗敬业意识的形成。

杨烈
东软集团Android软件工程师

敬业是一种态度

敬业这个话题几乎每年都会被提起，而且还会继续下去。也许，公司不消失，敬业就是一个永恒的话题。

在我的概念里，敬业是员工对企业和自己职业生涯的一种态度。这种态度是好是坏，主动权在员工，而企业与此却有着极其重要的关系。既然是敬业，这种态度就应该是一种积极的态度。它表现在以下方面。

第一，员工要有责任心。这种责任心不是简单地遵守公司的规章制度、不迟到不早退、老板交代的工作认真完成等这些表面的东西，而是，往大了说是发自内心地做事情为公司着想，小了说是全心全意地为自己所负责的事情着想。员工和公司是唇齿相依的，公司好，员工才能好。

第二，员工要有职业道德。大家都知道，工作中会有各种各样的诱惑，总结起来是金钱和权力的诱惑。如果员工为了一己私利而损害了公司的利益，那么他不但不敬业，而且有可能会毁了自己的职业生涯甚至触犯法律。

作为公司老板，应该努力做好以下几个方面。

第一，尊重员工。让公司充满人性化，让员工感觉到自己不是单纯来挣工资的，而是公司的一份子，我们在一个大家庭里，谁也不是局外人。你对别人够尊重了，别人才可能尊重你。

第二，营造一个宽松的环境。从事技术工作的人，经常工作很晚，会经常迟到，所以，这些方面宽容一些，他们工作起来会更卖力气。工作累时，出去散散步，看到工作园区美丽的绿化，心情会很放松、很愉悦。

第三，建立一套有效的激励机制。不要亏待员工，要在一定程度上体现出公平。可以很负责任地说，很多员工的不敬业甚至跳槽，与自己在公司中受到不公正待遇是分不开的。虽然世界上没有绝对的公平，但要让员工觉得，他的付出是有回报的，他的努力终究会得到公司的认可。

公司和员工是矛盾的两个方面，公司对员工的尊重和员工的敬业是相辅相成的。希望公司和员工一同成长，一同壮大，一同收获属于自己的成功，实现双赢。

寻找机遇 创造未来 CSDN 热门职位全新推荐

欢网科技

huan 欢网科技

欢网科技急聘：

- 高级嵌入式开发工程师
- 高级需求分析师
- Android TV 系统架构师
- 系统架构师
- 高级Java开发工程师

更多招聘信息和详细职位要求可登录欢网科技网站查询！
更多招聘岗位查询：http://www.huan.tv

简历投递：zhaopin@huan.tv
地址：北京市朝阳区劲松三区甲302号华腾大厦25层

欧特克软件（中国）有限公司

你一展身手的舞台，
打造美好未来

Autodesk

加入我们的队伍，以一种全新的方式实现设计理念。无论是设计可持续性建筑还是开发混合动力汽车，Autodesk软件都可助你一臂之力。作为全球领先的二维与三维设计、工程和娱乐软件提供商，我们致力于帮助打造出更美好的未来。

让我们共同创造未来!

更多职位信息请浏览网站：www.autodesk.com/careers
或直接将简历发送至：acrd.hiring@autodesk.com

搜狐畅游

搜狐 SOHU.com 畅游 ChangYou.com

游戏研发部3D程序专员

工作职责：
1. 3D框架下各种功能模块和算法的实现；
2. 研究、开发和优化3D引擎。

职位要求：
1. 诚信正直，具备高度的责任心；2. 计算机相关专业本科及以上学历；3. 两年以上的3D引擎开发经验；4. 精通C++，面向对象编程，Windows编程；5. 熟悉3D图形学原理，熟悉3D几何；6. 精通DirectX/OpenGL，精通DX9；7. 具备良好的算法和数学基础，优秀编码习惯和学习能力。

公司网址：http://www.changyou.com
E-mail：gamejob@job.cyou-inc.com

聚胜万合信息技术（上海）有限公司

聚胜万合MediaV依托强大的互联网广告平台，为广告主、网站主和战略合作伙伴提供专业的互联网营销服务，现热招云计算工程师、PHP工程师、数据处理应用工程师、产品经理等职位，邀您共谋中国数字营销大变局。如果您想在互联网广告领域一展鸿图，如果您拥有足够的经验和才能，MediaV期待您的加入。

MediaV

公司网址：http://www.mediav.com

方正国际软件有限公司

方正国际医疗卫生业务是中国首家医疗信息系统解决方案的提供商，经过16年的发展，已成为国内领先的数字化医院及区域医疗卫生网解决方案的提供商和服务商，并荣获“2011年度中国医疗领域最具影响力解决方案奖”，印证了方正国际医疗信息化领域的领军地位。

FOUNDER 方正

为适应业务的蓬勃发展，现在全国范围内诚征：

医疗信息化行业研发，工程项目，售前和销售精英

详细信息请登录智联招聘，前程无忧或中华英才查询，并投递相应邮箱！公司主页：www.founderinternational.com

厦门梦加网络科技有限公司

客户端程序员 （重点职位） 若干

至少一年相关工作经验，熟悉C/S架构，拥有良好的学习能力和独立解决问题的能力。根据游戏策划和主程序的要求，负责游戏客户端编程和维护工作。

MECHANIST GAMES

招聘要求：
1. 计算机相关专业，熟练掌握Java或C#编程语言，热爱游戏编程；
2. 使用过Unity3D引擎开发过游戏者优先；
3. 公司统一使用英文版本的技术软件，拥有大量外籍员工，英语口语较好者优先。

公司地址：厦门市思明区厦禾路666号海翼大厦A座11层
公司主页：www.mengjiagames.com
公司电话：0592-2661183骆小姐 **公司邮箱：**hr@mechanist.co

完美世界（北京）网络技术有限公司

完美世界（北京）网络技术有限公司（原完美时空）是中国领先的网络游戏开发商和运营商之一。公司成立于2004年，一直致力于创造优质的互动娱乐产业品牌，倾力打造拥有自主知识产权的高质量网游精品。

现诚聘

- ERP EBS 技术顾问
- flash(as3)开发工程师
- Java开发工程师
- 高级客户端开发工程师
- 游戏运维工程师
- PHP开发工程师
- 游戏服务器端软件工程师
- 游戏客户端软件工程师

职位详情请点击：http://special.zhaopin.com/bj/2011/wanmei03162823/index.htm
公司网址：www.wanmei.com　**电子邮箱：**zhaopin@wanmei.com

天津港保税区科技发展局

东软（天津）、展讯（天津）高薪诚聘IT英才

招聘职位：高级JAVA工程师，高级C++工程师，DSP工程师，芯片驱动工程师，应用软件工程师，驱动软件工程师，算法工程师

工作地点：

天津空港经济区（www.tjftz.gov.cn）

简历投递邮箱：resume@adm.tjftz.gov.cn

北京法国电信研发中心

Software engineer in the web service system development

Position requirements:

- Master degree in telecommunication, Compute Science OR equivalent
- Experience in J2EE, SOA, Spring, Hibernate, Struts, XML, REST, HTML5 development
- Experience in cloud computing development, such as SaaS, PaaS, etc.
- Strong programming skills, at least 3+ years experience in Java software development
- Familiar with the software development methodology
- Familiar with networking technology
- Good English communication skill
- Good team working spirit and communication skills, and hard working as well

Contact：hr@orange-ftgroup.com.cn

北京神舟航天软件技术有限公司

神舟软件是由中国航天科技集团公司控股的国家重点高新技术软件企业，公司业务领域涵盖工业软件与服务、电子政务与公共服务、基础软件、系统集成、航天文化创意等方面，拥有一批具有自主知识产权的实用型软件产品，产品与服务遍及国防、军工、政府、机械制造、医疗卫生、教育等多个领域。

因发展需要诚征人才，我们会提供广阔的发展空间和完善的保障。

- 高级PLM业务架构师
- 高级Java系统架构师
- 高级Java开发工程师
- 项目经理\产品经理
- UI架构师
- 中级Java开发工程师

更多职位信息可浏览：http://www.bjsasc.com/rlzyavidm.html
简历投递：zhaomengyi@bjsasc.com　**联系人：赵梦祎**

爱立信（中国）通信有限公司

爱立信专注于通信研发130余年，始终屹立无线通信领域行业领先的地位。而其多媒体软件系统更是独树一帜，成为爱立信的主要战略方向。爱立信公司多媒体事业部上海的研发中心正肩负这样的使命，其中的TV（包括IPTV，WebTV，CableTV和MobileTV）产品线更是战略重点，为此我们诚邀您加盟。

我们这次主要招聘三类职位：

1.MobileTV/WebTV的软件开发职位
2.TV解决方案的集成和测试
3.TV机顶盒的集成和测试

广州瀚信通信科技股份有限公司

招聘职位：

- C/C++ 系统架构师
- C/C++ 网络开发工程师
- Silverlight/WPF 前端交互设计师
- C# 开发工程师
- 软件测试工程师

详细信息请登录CSDN网站查询！

请以“职位+姓名+CSDN”作为邮件标题，
将简历发送至：yfhr@hantele.com　**公司主页：**www.hantele.com

云计算平台管理的三大利器

Nagios、Ganglia和Splunk

文 / 杨俊华

综合利用Nagios、Ganglia和Splunk搭建起的云计算平台监控体系，具备错误报警、性能调优、问题追踪和自动生成运维报表的功能。有了这套系统，就可轻松管理Hadoop/HBase云计算平台。

云计算早已不是停留在概念阶段了，各大公司都购买了大量的机器，开始正式的部署和运营。而动辄上百台的性能强劲的服务器，为运营管理带来了巨大的挑战。

- 如果没有方便的监控报警平台，对于管理员而言犹如噩梦，每天都将如救火队员一样，飞快地敲击键盘，用原始的Unix命令在多台机器中疲于奔命。
- 如果没有好的日志管理平台，对于开发者Troubleshooting更是一件泪流满面的事情。
- 而如果你是运维团队的总负责人，简洁清晰的Report则非常重要。Stakeholder们动不动就可能问起系统的SLA、机器的利用率等诸多问题，毕竟，公司为此投入了巨大的资金和人力。

朋友们，当我们管理起公司寄予厚望的云计算平台时，当我们面对如此多充满挑战的实际问题时，该怎么办？

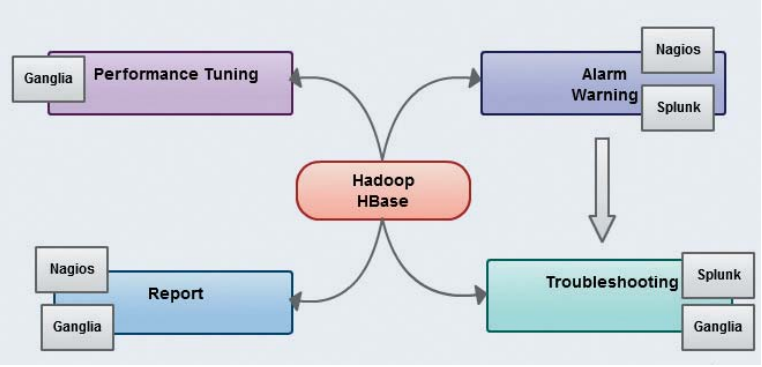


图1 云计算平台监控架构

概述

我们在搭建趋势云计算平台时，遇到了很多的问题和挑战。开始搭建时，第一次来了那么多性能强劲的机器，我们在感到兴奋的同时，也不免有些顾虑。大家坐在一起讨论，问题就列了满满一白板。

- 出了问题怎么办，有没有预警机制？
- 有没有可视化的管理界面？
- 管理平台需要自己开发吗？开发难度有多大？
- 有没有开源的管理工具？
- 那么多日志分布在各个机器上，有没有更有效的方法管理？
- 能否生成好的报表？
- 机器宕机，管理员能否收到短信通知？
- 如何做性能调优？
- 扩容升级时，能否给出依据？

……

带着这些问题，我们开始了自己的云计算平台管理和运营之旅，一路走来，收获颇丰。现在基本上形成了如图1所示的一整套云计算平台监控体系。

在这个系统中，我们综合利用了Nagios、Ganglia和Splunk，搭建起云计算平台监控体系，使其具备错误报警、性能调优、问题追踪和自动生成运维报表的功能。有了这套系统，我们终于能够轻松管理Hadoop/HBase云计算平

台了。接下来将简单介绍它们的特点和功能。

Nagios：云计算平台的智能报警器

总不能天天盯着机器看吧，因此我们首先关心的是机器的监控与报警。最理想的境界是：如果机器出故障了，我能第一时间处理；如果机器没有问题（最好永远没有问题），我能去喝茶、钓鱼和睡大觉。

发现机器有没有问题，对我们而言不是什么难事。写个脚本，Ping一下IP，Telnet每台机器的Service端口，如果增加了新机器就改改配置即可。但这样也太原始了吧，可视化效果差，不好维护，没有层次，不好管理，出不来报表，总不能老是用Excel人工写报表吧。有没有更好的方法呢?

有，你可以用Nagios。

Nagios是一个可运行在Linux/Unix平台之上的开源监视系统，可以用来监视系统运行状态和网络信息。Nagios可以监视所指定的本地或远程主机以及服务，同时提供异常通知功能。

Nagios可以提供以下几种监控功能。

- 监控网络服务（SMTP、POP3、HTTP、NNTP、Ping等）。
- 监控主机资源（处理器负荷、磁盘利用率等）。
- 简单的插件设计使得用户可以方便地扩展自己服务的检测方法。
- 并行服务检查机制。
- 具备定义网络分层结构的能力，并使用“parent”主机定义来表达网络主机间的关系，这种关系可被用来发现和明晰主机宕机或不可达状态。
- 当服务或主机问题产生与解决时将告警发送给联系人（通过电子邮件、短信、用户定义方式）。
- 具备定义事件处理功能，可以在主机或服务的事件发生时获取更多问题定位。
- 自动的日志回滚。
- 可以支持并实现对主机的冗余监控。
- 可选的Web界面用于查看当前的网络状态、通知和故障历史、日志文件等。

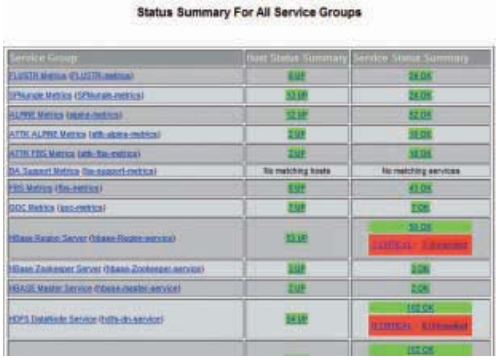


图2 SPN 后台运行的所有Service的当前状态

Nagios最好用的地方就是它将这些每天管理员做的工作自动化，你只需设定好要监听的端口即可，它会默默地工作，帮忙定时地去检测服务端口的状态，一旦发现问题，会及时发出报警。报警可以是电子邮件也可以是手机，从而使得管理员第一时间就能收到系统的状况。

Nagios的报表功能也很强大。管理员可以很容易地得到每天、每周和每月的Service运行状况。

如图2所示，红色部分清楚地标注有问题的机器，点开链接，就可以得到有问题机器的情况。虽然在HBase中，几台Region Server宕机不会对整体服务产生大的影响，但多少会影响到系统的Performance。而且，如果某几台Region Server频繁宕机，对整个系统的稳定性也会产生不好的影响。有了Nagios，我们可以快速定位有问题的机器，及时地将一些机器移除出HBase系统，待调整好了再上线运行，以保证系统的稳定性。

现在，Nagios已经成为了很多公司必备的监控工具。只需要简单地配置，就可以实现强大的功能，将管理员从日常烦琐的工作中解放出来。

有了Nagios，哪怕就是管理上千台机器，也不会手忙脚乱，而是有一种统领千军、运筹帷幄的感觉。

Ganglia：看到云计算平台的方方面面

Nagios的确不错，但你是不是真的可以喝茶、钓鱼、睡大觉呢？显然还不行。有了Nagios，你基本上可以做个优秀的救火队员，能

在事发第一时间到达现场、处理事故。但如何防患于未然，真正做到运筹帷幄、游刃有余呢?

我们需要更加精确的数据，能够看到云计算平台的方方面面，能根据这些数据，做出性能调整、升级、扩容等的决策，从而保证Service能够满足不断增长的业务需求。

这时候，你需要Ganglia。

Ganglia是UC Berkeley发起的一个开源实时监视项目，用于测量数以千计的节点，为云计算系统提供系统静态数据以及重要的性能度量数据。Ganglia系统基本包含以下三大部分。

Gmond：Gmond运行在每台计算机上，它主要监控每台机器上收集和发送度量数据（如处理器速度、内存使用量等）。

Gmetad：Gmetad运行在Cluster的一台主机上，作为Web Server，或者用于与Web Server进行沟通。

Ganglia Web前端：Web前端用于显示Ganglia的Metrics图表。

Hadoop和HBase本身对于Ganglia的支持非常好。通过简单的配置，我们可以将Hadoop和HBase的一些关键参数以图表的形式展现在Ganglia的Web Console上。这些对于我们洞悉Hadoop和HBase的内部系统状态有很大的帮助。

在Hadoop的conf文件夹下面，找到hadoop-metrics.properties，配置好Ganglia的Server即可。这里要注意，Ganglia 3.0和Ganglia 3.1的区别，它们使用了不同的class。

```
dfs.class=org.apache.hadoop.metrics.
ganglia.GangliaContext31
dfs.period=10
dfs.servers={Ganglia_Server}:8649
```

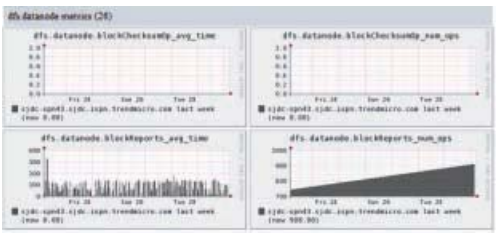


图3 Hadoop其中一个DataNode的Metrics

有了这些图表，Hadoop和HBase就不再是一个黑盒。无论是Hadoop的Namenode、Datanode，还是HBase的MasterServer、RegionServer任何时刻的情况，都会一目了然。由于图标的跨度可以是小时、天、月甚至是年，这样，就可以非常方便地定期生成周报、月报和年报。同时，根据图中Metrics的状况，我们可以通过调整参数、增加内存和硬盘、增加机器等的方法调整单个机器或者整个Service的性能。

Nagios 最大的问题在于不能洞悉到Service内部的状况。像Hadoop、HBase这样的分布式系统，一个节点的故障并不等于整个Service的故障，影响的只是Service的性能。所以，在测定Service的SLA时，我们不能以某一台机器的故障作为Service故障的评判标准。比如在我们的HBase SLA的设定上，我们定义了HBase Service完全不能工作的评判标准如下。

- Master Server 联系不上。
- 所有RegionServer 都无法联系上。
- -ROOT- 表无法访问。
- .META. 表无法访问。

那么，我们就可以根据这个规则定义SLA，通过定期调用HBaseAdmin相应API，将测试的结果发给Ganglia。采用同样的方法，我们还可以自定义一些规则，监视HBase Master、Zookeeper等的情况。

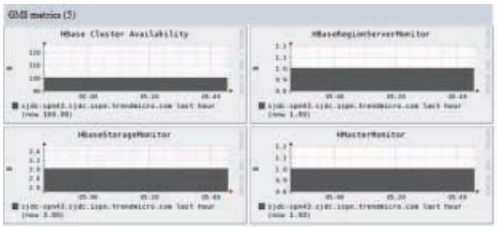


图4 Ganglia对Hadoop/HBase使用情况的监测

通过这些方法，我们完全能够针对Hadoop/HBase使用的实际情况，做出Service级别而不是机器级别的监控系统并生成报表。

此外，Ganglia还可以通过Server反馈回来的Load信息，给出各个机器的Load情况，给我们做升级和扩容提供依据。

如图5所示，Ganglia分别会用不同颜色，标注出当前时刻的机器Load分布情况。如果Load过重，就应该检查机器的具体使用情况。

Ganglia的安装配置，可以参考：http://www.spnguru.com/?p=604。

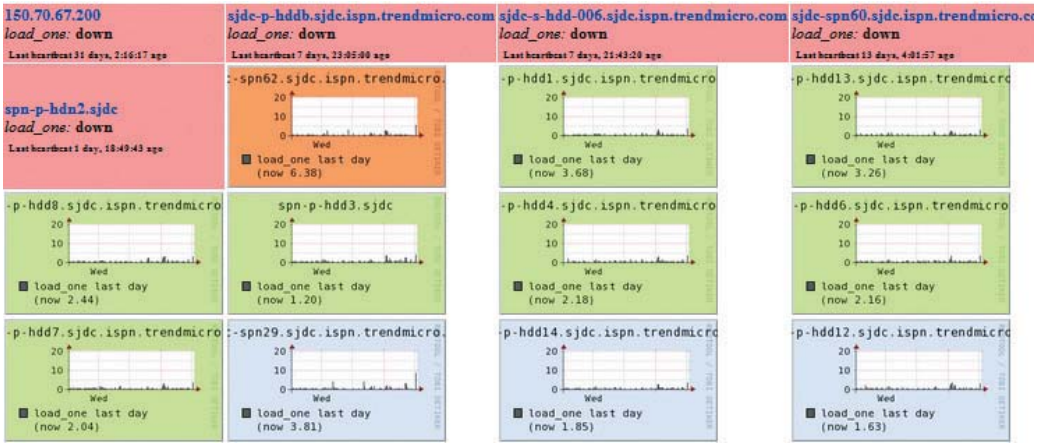


图5 HBase Cluster Load Metrics

Splunk：像查Google一样查日志

有了Nagios和Ganglia，算是成功了一大半。作为一名优秀的管理员，我们需要具备一定的Troubleshooting能力，对一些常见的问题能给出解决方案。那么，对日志的分析就必不可少。

但Hadoop/HBase的日志分布在各个机器上面，而日志之间关联性强。Client端的错误有可能是Region Server引起，而Region Server的错误有可能是Zookeeper导致。有没有一个统一的日志管理平台呢?

众里寻它千百度，蓦然回首，我们找到了Splunk——日志界的Google。

很遗憾，Splunk不是开源的，但它的免费版本提供每天500MB日志索引。如果数据量较小，通过定义好Log的级别，基本上也能满足需求。但对于数据量较大的公司，就有些捉襟见肘。

Splunk支持AdHoc的日志搜索，而且可以与Nagios配合使用。比如Nagios报警某台RegionServer端口不可达，我们收到Notification后，登录Splunk，直接搜索shutdown和host名称，找到RegionServer退出的日志。点击详细信息，分析日志，就能快速定位问题。如图6所示。

对Hadoop和HBase有了进一步了解后，我们可以利用Splunk实时检测日志中的关键字，定义关键字规则，如监控“shutdown”、“quit”、“ERROR”、“Zookeeper Session Expired”等，一旦出现，利用Splunk的Notification功能，发出邮件通知管理员，管理员通过Splunk定位问题，就可以在系统真正出现问题之前，对系统进行调整，防患于未然。具体Splunk的设置，可以参考：http://www.spnguru.com/?p=122。

图6 Splunk与Nagios配合使用进行日志搜索

总结

搭建一套云计算平台，强大的监控管理系统是必不可少的。当然，任何工具都不是万能的，在实际维护过程中，我们也发现，Nagios和Splunk经常出现误报，如果规则定义得不好，大量的警报邮件如潮水一样涌来，反而掩盖了真正的问题。可以说，在云计算平台的运维管理上，没有一劳永逸的事情，随着规模的不断增大和应用的不断多样化，需要大家不断地实践和总结。

杨俊华

趋势科技研发中心资深开发工程师，2009年至今一直从事Hadoop和HBase开发和运维工作，关注Hadoop开源社区的发展。

责任编辑：董世晓（dongsx@csdn.net）

在路上——图形数据库

文 / 李祎

在作者看来，分析社会关系这类复杂图壮结构的海量数据，使用图形数据库（Graph DataBase）是最好的选择。

作为NoSQL数据库的一种，图形数据库（Graph DataBase）很长时间都局限在图论的学术圈子里，在业界使用得并不广泛。直到近十年电子商务的业务模式逐步成熟，电商们需要对用户在其网站的购买行为进行分析、发掘用户潜在喜好的商品，开始大量使用到了图论相关的数据挖掘算法，图形数据库才从实验室走进业界。特别是随着Facebook为代表的SNS网站兴起，对上千万的用户行为和关系数据进行挖掘，发掘其商业价值越发成为迫切的需求和工程难题。工程师们发现使用图形数据库来存储和计算这些海量数据会更为高效和方便，这极大推动了图形数据库的发展和它在NoSQL世界的知名度。现在比较著名的图形数据库有Twitter的FlockDB，以及开源项目中的Neo4j和Infogrid。

图形数据库的基础概念

■ 点（Node）表示一个实体，例如人、商品或者一个账户。

■ 边（Edge）也称做关系（Relation），表示点和点之间的连接关系。通常边是有方向性的，例如用户A购买了商品B表示为A→B；如果用户A和用户C互相都认识，这种关系就是双向的，表示为A⟷C。

■ 属性（Properties）表示点和边所附带的信息，例如用户的属性可能是年龄30岁、爱好篮球等。需要注意的是，每个点的属性是动态的，例如同样是用户A和C，A有属性“年龄30岁”，C却没有，但它却包含属性“职业工程师”。

■ 把点、边、属性融合在一起，就可描述出一个图（Graph）。图1描述了模拟《黑客帝国》的小型社会图（Social Graph）。

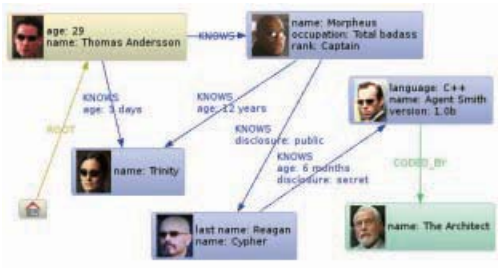


图1 《黑客帝国》的小型社会图

在图1中，每个头像都代表了一个点。边用蓝色箭头表示，箭头方向表明两者的关系，边也可拥有属性，例如Morpheus认识Trinity包含属性age:12 years。当千万个点和边被关联起来，这张社会图就被无限放大，构成一张描述整个虚拟社区的人际关系图。图形数据库就是用来存储这张海量数据关系图的唯一的最优工具。

传统的关系型数据库和其他NoSQL数据库不能最优化地存储上述社会关系数据。一方面，图形数据库中每个点包含的属性不同。如果是在关系型数据库中建立表结构，需要有许多列的大型表，其中很多行的列是空的（稀疏表），查询时需要联结大量表、使用深度嵌套的SQL，这也导致了较低的性能，不适合高性能查询。另一方面，图形数据库针对图算法提供了很多高效的操作特性，这也是它在图计算中表现优异的原因。

■ 遍历（Traversal）：不同于其他NoSQL数据库，图形数据库支持一系列图遍历的操作。如图1中，找到Thomas和Architect之间的最短路径，通常遍历操作需要传入两个参数，一个是起始点，另一个是使用何种遍历规则。遍历返回的结果，可能是零个、一个或多个点的集合。

■ 事件（Events）：这与关系型数据库中的触发器比较类似，当某个事件发生时触发某个操作。不同之处在于，关系数据库不能将事件通知到数据库外的应用程序；而图形数据库提供了可

编程的监听接口来供外部程序监听各类事件。如图1中，外部程序可以注册事件监听器（Event Listener），当Smith的version属性从1.0变到5.0时，图形数据库将这个事件通知事件监听器。

■ 索引（index）：在关系型数据库中，对某张表进行过多的索引，当数据量大增时，效率就极其低下。而在图形数据库中，对所有点的属性都可以建立索引，即使是千万级别的数据量，也能提供高效的查询。

图形数据库在数据结构、存储、遍历以及查询方面都有别于其他NoSQL数据库。因此，对于社会关系这种复杂图状结构的海量数据进行分析时，图形数据库是最佳选择。不过现在业界还没有一款称得上成熟的产品级应用，国内实际使用过图形数据库的项目更是少之又少。与其苦苦等待，不如自己动手搭建一个可用的图形数据库。下面以我们开发的图形数据库Skull DB为例，介绍在实际项目中如何实现和使用图形数据库。

Skull DB介绍

Skull DB实现了上述图形数据库的所有特性，不过在具体实现的数据结构上略有差别。在Skull DB的数据结构定义中（见图2），Node类代表点；id用来标识Node在图中的唯一性；size表示该Node包含的边总数。Relationship类代表另一个点和另一点的边，表示两个点的唯一关系；rel_id表示另一点的id；rel_value字段用于存储与边相关的任何属性（例如收听关系中被重复收听了多少次等信息）；modify_time表示边建立和修改的时间。RelationshipList代表一个Node和它所有的Relationship集合，它内部由一个按Relationship的modify_time字段倒序排列的指针链表（LinkArray）组成。

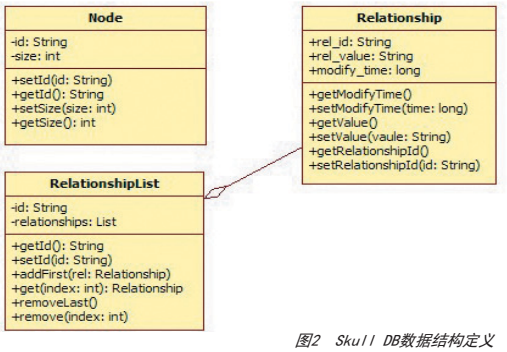


图2 Skull DB数据结构定义

下面以《黑客帝国》中的收听关系图（如图3）为例，详细介绍如何使用Skull DB。

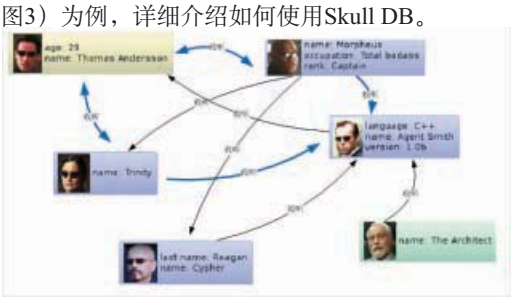


图3 《黑客帝国》的收听关系图

在图3中，Thomas表示为一个点：

```
Node thomas_node = new Node("Thomas");
```

Thomas收听了Trinity和Morpheus，其收听关系可表示为：

```
RelationshipList  followingList = new
  RelationshipList("Thomas");
Relationship following_trinity = new
  Relationship("trinity', "1");
// "1"代表Relationship 的value字段，在这里
用"1"表示收听了1次。
followingList.add(following_trinity);
Relationship following_morpheus = new
Relationship(“morpheus”, "1");
followingList.add(following_ morpheus);
```

详细步骤如下所示。

■ 将用户收听的行为数据导入Skull DB。一般我们拿到的数据是从生产环境导出的txt文件，使用Skull DB提供的命令行工具txt2skull，将其导入到Skull DB中。

命令行：

```
txt2skull.sh  family  txtfile
```

第一个参数family，类似于BigTable中的column family，指定数据存储到那类数据集中。由于是用户收听关系，family用following表示。

第二个参数txtfile，指明需要导入的用户好友关系文件位置。这里文件名是user_following.txt。

命令行：

```
txt2skull.sh  following  user_following.txt
```

接下来就可以使用Skullclient对SkullDB进行访问，获取Thomas的收听关系：

```
RelationshipList  thomasRelations =
  SkullClient.getRelationships("following",
  "thomas", Direction.POSITIVE);
for(Relationship relation : thomasRelations)
{
  printRelation (relation)
}
```

结果是：Trinity, Morpheus。

这里Direction.POSITIVE指定了查询Thomas收听的人，如果想查询都有谁收听了

Thomas，使用Direction.REVERSE：

```
RelationshipList  thomasRelations =
  SkullClient.getRelationships("following",
  "thomas", Direction.REVERSE);
for(Relationship relation : thomasRelations)
{
  printRelation (relation)
}
```

结果是：Agent Smith。

■ 通过SkullClient查询，编写一个推荐算法，查询Thomas收听的人还间接收听了谁。

```
//allMap中包含了Thomas收听的人所有的收听列表。
Map<String, RelationshipList> allMap =Skull-
    Client.getMultiRelationships("following",
    aRelations.getIds(), Direction.POSITIVE);
//遍历所有列表，累计重复次数最高的Relationship。
SortedMap<String, Integer> topMap = new
    TreeMap<String, Integer>();
for(RelationshipList relationList:allMap.values())
{
  for(Relationship relation : relationList)
  {
    if(topMap.get(relation.getId())!=null)
    {
      int repeats = topMap.get(relation.
        getId()).int-Value() + 1;
      //累计重复被收听ID的出现次数
      topMap.put(relation.getId(),
        new Integer(repeats));
    } else {
      topMap.put(relation.getId(),
        new Integer(1));
    }
  }
}
System.out.print("你关注的人间接关注了:" +
  topMap.lastKey()); // 打印重复次数最高的ID
```

结果是：Agent Smith。

■ 为了阻止数据库中的数据无限制增长，需要删除收听时间大于1年的过期数据。这里需要实现RelationEventListener接口。当发现Node添加了新的Relationship时，遍历RelationshipList，删除收听时间大于1年的Relationship。

```
public class OldRelationCleaner implements
RelationEventListener {
  private fina long one_year = 365 * 24 *
60 * 60 * 1000;
  public boolean EventReceived( Event
    event, EventData data ) {
    String node_id = event.getNode().
getId();
    String family = event.getFamily();
    RelationshipList relationships =
      skullclient.getRelationships(family,
      node_id, Direction.POSITIVE);
    for (Relationship relation : relationships)
    {
      if (System. currentTimeMillis()-
        relation.getModifyTime() > one_year )
      {
        skullclient.removeRelationship(family,
          node_id, relation);
      }
    }
  }
}
```

在程序中注册收听事件：

```
EventManager manager = EventManager.
  getInstance();
manager.registerRelationEventListener
  ("following",new OldRelationCleaner());
```

当Thomas收听了Architect后，在EventManager上注册的RelationEventListener被触发，遍历代表Thomas所有收听关系的RelationshipList实例Relationships，找到收听时间大于1年的Relationship并删除。

Skull DB实现

简单地说，Skull DB是一个以Memcache作为数据缓存、Solr和Lucene负责数据物理存储和索引、MemcacheQ来保存事件的集成系统。由于Solr和Memcache都有很好的横向扩展性，因此Skull DB适合作为用户行为和关系等的数据仓库，提供高效的查询和运算，如图4所示。

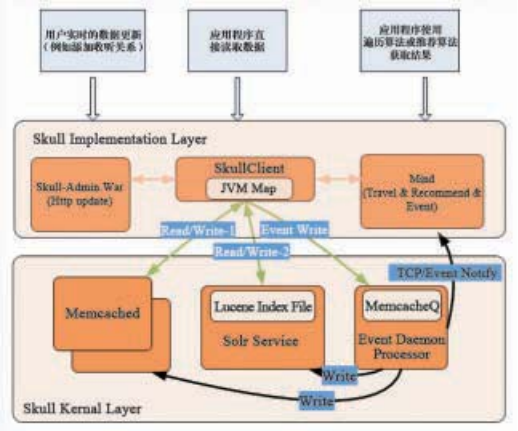


图4 Skull DB架构

客户端SkullClient封装了所有对Skull DB的访问，它使用多级缓存和多线程从Skull DB获取数据。所有对Skull DB中Node和Relationship的创建、读、写以及删除操作，都通过SkullClient完成。当调用读接口时，SkullClient先访问Memcache，如果找不到数据，再发送HTTP请求从Solr获取。当调用写接口时，SkullClient将被修改的Relationship调整到RelationshipList列表的第一位，更新Memcache中数据的同时，异步发送HTTP请求将修改后的Lucene文档发送给Solr，并发送异步TCP请求，将写事件通知Event Daemon模块。

Skull-admin.war模块可部署在任何Web容器（例如Tomcat）内，提供REST接口对Skull DB进行读写操作。它还可以通过JMX控制台进行以下操作：实时获取Memcache状态（如图5）、查询Solr统计数据、管理Event Daemon内部队列、查询Skull DB内部数据、进行数据导入和导出等。

mind_jp

/192.168.10.37:11211

Memcache Server version:	1.4.4
Process id of this server process	9735
Number of seconds this server has been running	2774185
Accumulated user time for this process	448.273852 seconds
Accumulated system time for this process	588.116592 seconds
Number of open connections	19
Total number of connections opened since the server started running	118
Number of connection structures allocated by the server	21
Cumulative number of retrieval requests	31686371
Cumulative number of storage requests	29504792
Number of keys that have been requested and found present	45%
Number of items that have been requested and not found	55%
Total number of items stored by this server ever since it started	29504792
Used memory size by this server ever since it started	468 Mega Bytes
Average size of item stored by this server ever since it started	16 Bytes
Total number of bytes read by this server from network	85474 Mega Bytes
Total number of bytes sent by this server to network	8773 Mega Bytes
Number of bytes this server is allowed to use for storage.	512 Mega Bytes
Number of valid items removed from cache to free memory for new items.	8659539

图5　实时获取Memcache状态

Mind模块封装了对Skull DB的遍历和事件监听等方法，同时还包含一些常用的推荐算法。该模块对Skull DB的读写通过调用SkullClient来完成。当有Node和Relationship被创建或更新时，SkullClient调用异步线程将事件发送到Event Deamon模块，事件被直接保存到MemcacheQ中。事件后台进程负责处理保存在MemcacheQ中的事件，更新Memcache和Lucene索引中的Node和Relationship对象。处理完成后，事件后台进程将事件再发送给Mind模块的EventManager，通知所有注册事件的订阅者。

性能分析

我们使用一个新搭建的Skull DB进行性能测试，用20个并发线程向其插入用户好友关系。结果如图6所示，X轴是测试执行的时间间隔（精确到秒），Y轴代表响应时间。本次测试总共插入109645条Relationship记录，耗时6分28秒，即每个线程平均85毫秒完成一次更新。其中有两个比较陡的响应时间波峰，是因为Lucene在进行索引写文件操作时，I/O会受到一些影响。

接下来使用SkullClient对Skull DB进行查询。本次查询使用20个并发线程，一共遍历了15211个用户的好友关系。由于前面已经插入了109645条Relationship记录，所以每个Node平均包含有7个Relationship。结果如图7所示，X轴是测试执行的时间间隔（精确到秒），Y轴代表响应时间。本次测试共查询了15211个用户的好友关系，耗时12秒，即每个线程平均15毫秒完成一次查询。由于Relationship数据都被缓存在Memcache中，所以响应时间的趋势是前高后低。

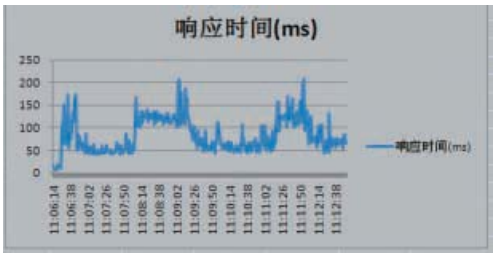


图6　Skull DB插入操作性能测试

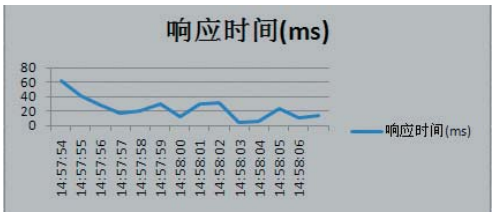


图7　Skull DB查询操作性能测试

总结

在移动微博网站，我们用Skull DB存储用户的关系数据、评论转发数据、登录IP数据等，配合MapReduce框架来计算亲密度、关系度、热度等指标，此外还用于BI分析统计，保存用户的标签信息用于标签分类，发掘潜在用户的商业价值。我们之所以自己实现图形数据库是由于数据存储方案无法胜任高并发计算、高I/O吞吐效率、对用户关系进行两步以上遍历的需求。另一方面，我们对数据一致性要求不高，即使有少数数据不一致也不影响整体。在实际项目中，Skull DB还存在Lucene索引效率需要加强，以及数据安全不高等方面的问题。目前我们仍在不断完善它。Skull DB的成功不远了，我们还在路上。

李祎

139社区系统架构师，负责海量数据存储访问、推荐引擎架构以及图形数据库开发。关注移动和互联网领域的新技术和新应用，曾主持开发了139网站的用户平台、社区平台、监控平台。

责任编辑：董世晓（dongsx@csdn.net）

探究架构方法论实践之路

记者 / 付江

决定软件系统质量最重要的因素是软件架构。如果把架构比作骨架，那么方法论就是它的血液，其重要性自不必说。既然架构方法论如此重要，那么它究竟是什么？一百个架构师会得出一百种不同的答案。一些人会觉得架构全靠经验，没有什么方法论，更有甚者，认为架构方法论简直就是一套神秘的形而上学的东西。事实上，它并不神秘，它有章可依，有迹可循。在IBM看来，架构方法论就是通过预定义的一套可重用的内容和流程，帮助架构师系统设计架构的方法，而且这个方法是可以学习并传承的。

从另一个角度，它也和许多其他方法论词汇一样抽象，无法真实触及实体，但影响力却无处不在。随着IT系统越来越复杂，业界对IT系统如何正确构建、逐日提升更加关注，国内越来越多的公司开始研究和实践相关理论，个别公司甚至组建了专门研究和实施方法论的团队。

作为全球最大的IT服务商，IBM有着全世界最系统、最完整的方法论，不是单单的面向硬件或者中间件方面的架构，而是涵盖从企业架构、业务架构、应用架构、信息架构、基础架构乃至集成架构等都有详细的方法论。

他山之石，可以攻玉。日前，就架构方法论在国内的现状以及国内企业应该如何实践，《程序员》杂志记者与IBM软件集团大中华区合作伙伴技术支持总经理王小虎进行了一次对话。

尽早营造出重视架构方法论的氛围

据王小虎观察，国内经过实战锻炼出来的技术英雄众多，不管名片或职称上是否印了架构师的头衔，实际上都已经在做大量架构师的工作了。很多时候，中国工程师能凭借个人天赋和经验将项目带向成功，虽然一将功成万骨枯，其经验往往带着血的教训。不过随着项目环境的变化，过去成功的经验往往并不适用，这时候往往会给项目带来巨大的风险，更为可怕的是随着英雄的离去，项目就进展不下去或公司一条线倒下的情形比比皆是。究其原因都是因为缺乏统一的标准和方法论来指导，难以将经验系统化地积累起来并科学地使用。

与之相对的，国外同行已经通过大规模普及架构方法论应用，批量“生产”出一批批的架构师。这些架构师未必每个人都有天赋，但通过系统培训，一步步按照标准的方法论走下来。最后设计出来的架构文档形式貌似都差不多，但考虑都比较周详和全面，绝大多数项目都能保证成功，起码，很难因为技术问题而失败。而且，更重要的是这些宝贵的资产和标准的代码模块可以很好地在公司传承下去。

王小虎建议，任何将软件视为核心资产的公司都应该尽可能早地营造出重视架构方法论的氛围，以及建立稳定的架构师团队和架构师培养计划。优秀的架构方法论从单个点上来说可以让架构师更合理地设计方案，帮助项目成功，对于公司来说还可以让不同的团队用一种语言和方法办事，传承公司一体化的思维。反过来看，他所见过的因缺乏架构方法论指导而导致项目后期陷入困境，甚至公司因此破产的情况让人触目惊心。

单纯从技术角度上看，即使在理论上，都很难通过架构重构等增量技术来将一个糟糕的架构改良为一个优秀的架构，在实践中就更难了。后期重构架构的成本将非常高昂，以至于超出了绝大多数管理者所能接受的程度，所以往往不得不推倒重来。如果早期就重视方法论，并将其运用到工作中，以此为基础，一个好的架构可以在成本很小甚至零成本的时候就建立起来，由此减少的项目执行过程中的重新设计和返工将带来良好的投资回报。从开发者的角度看，由于架构的各个层次和元素设计得更好，应用开发人员不需要了解那么多其他层次和元素的技术细节，可以更关注所构建模块

王小虎：尽可能早地营造出重视架构方法论的氛围

的解决方案，开发出的软件质量也更高。如果已经建立了一个良好的软件系统架构，就可以持续地重新评估，并做出必要的完善和重构。

IBM所提倡的架构方法论所遵循的基本原则，包含了四个关键词：预定义、前后连贯逻辑严谨、迭代式、多重视角。具体说来，就是要以活动来驱动设计流程，而不是工作件（Work Product）。每个工作件都需要多项输入，同时自身也是其他工作件的输入，一旦获得工作件信息要立即记录。工作件处于迭代过程中，常常会经历若干次的修改和优化。部分工作件会有多重的视图，同时部分工作件只针对特定的读者。每一个技术决策都可以被回溯到需求中的某一部分。王小虎强调，所有这一切的最终目的都只有一个，就是设计优良的架构并尽可能地进行资产重用，这是架构方法论所追求的软件系统价值最大化的最重要的原则。

分阶段推进架构方法论落地

从国内多数公司现状来说，虽然从老板的角度都想将架构方法论推行起来，但现实问题是，迫于财务压力和人才短缺，公司里真正有能力去做这些研究的技术精英都已经被抽去做业务项目，导致架构方法论的推进有限。对此，王小虎认为，任何公司都不应该仅依赖1~2个能独当一面的高手来解决所有问题。推广方法论的时候不要操之过急，首先可以成立一个架构方法论研究的团队，可以是实体部门也可以是虚拟的团队，此外从公司业务和组织架构上作出调整以适应知识积累及共享机制；其次，王小虎建议将眼光放得长远一点，分阶段地实现架构方法论，可以在部分团队中挑一些重要的方法环节进行统一的实施，通过阶段性成绩去说服上级和业务部门主管，让他们看到好处，以获得进一步的支持。可喜的是，国内很多有一定规模的公司都开始重视和应用架构方法论了。

在和王小虎交流的过程中，记者深切地感受到王小虎胸怀中国软件产业的责任和对国内架构师的殷殷期望。为了帮助国内架构师团队的成长，在他的带领下，2010年8月起IBM就在国内启动了IBM中国渠道大学软件架构师培训计划。这次IBM将以往视为传家之宝的只是针对IBM内部员工的架构师培训，面向合作伙伴推出，而中国也成了全球范围内首批享受IBM架构师培养计划的国家。到2011年5月，首批通过第一阶段认证的26位架构师，获得了“助理架构师”证书。

海量数据处理生态系统

文 / 杨栋

O'Reilly Strata Conference议程主席Edd Dumbill的*The SMAQ stack for big data*一文从架构上全面精辟地展示了海量数据处理的生态系统。本文以这篇文章为基础，对海量数据处理生态系统做了进一步展示和分析。

SMAQ是Storage、MapReduce And Query的缩写，意为利用分布式存储、分布式计算并提供基础查询来实现对海量数据（Big Data）的处理。SMAQ生态系统将海量数据处理领域推向一个新的高度，支撑起数据驱动为核心的产品和服务的新时代。作为O'Reilly Strata Conference和O'Reilly Open Source Convention的议程主席，Edd Dumbill在2010年9月发表了一篇关于SMAQ系统的文章——*The SMAQ stack for big data*，从架构上全面精辟地展示了海量数据处理的生态系统，网络上也有很多转载和翻译。本文基于这篇文章，对海量数据处理的生态系统做了进一步展示和分析。

海量数据处理和传统数据处理方式有所不同，是一项更加复杂的工作，原因大致包括以下三个方面。

- **容错问题**：海量数据规模过大，数据存在未知状态，例如PB级的处理数据，格式化后可能依然存在特殊情况，因此如何控制海量数据处理的误差，以及提高数据处理系统的容错性是个棘手的问题。
- **资源问题**：海量数据处理对软硬件资源要求较高，资源需求较大，处理时需要大量的CPU、内存、网络、I/O等资源，如何节约成本，使用最少最有效的资源，保证资源使用的稳定性是关键问题之一。
- **时效问题**：海量数据处理流程长、耗时长，如何将处理结果及时有效地反馈是重中之重。

不论是Web搜索引擎、社交网站，还是移动应用，在科学计算领域，每天都有超过PB级的数据产生。作为海量数据处理的先行者，Google发明了简单高效的并行数据处理模型MapReduce，以及适用于海量存储的GFS文件系统和BigTable索引系统。Yahoo!基于Google工作创建的开源系统Hadoop及其周边开源项目，孵化出了整个海量数据处理的生态系统，这样的生态系统已在国内外各大公司形成，像Yahoo!、Facebook、百度、淘宝、腾讯等。

正如LAMP（Linux + Apache + MySQL + PHP）生态系统改变了互联网应用开发领域一样，SMAQ支撑了数据驱动为核心的产品和服务的新时代。如图1所示，引用CIDR2011会议论文Starfish中的一幅图片，从Hadoop生态系统的角度来描述海量数据处理的生态系统概貌。

海量数据处理系统的输入数据来自多个不同的源系统，它们一般都是面向用户的OLTP（On-Line Transaction Processing）系统或者是记录用户行为的日志系统，比如Key-Value存储系统、数据库或者其他中间件服务。OLTP是面向交易的处理系统，其基本特征是用户的原始数据可以立即传送到计算中心进行处理，并在很短的时间内给出处理结果。这样做的最大优点是可以即时地处理输入的数据、及时地回答，因此也称为实时系统。数据从OLTP系统流向分析系统，处理后的部分结果近实时地导入面向用户系统，如果是TB级数据，导入周期或许会持续到天级。处理海量数据的核心是Hadoop系统，包括MapReduce计算框架、HDFS或HBase等存储服务。分析海量数据的命令可

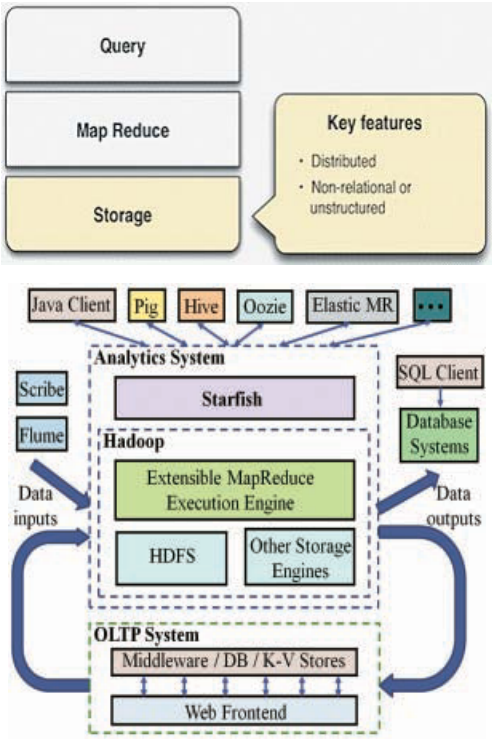


图1 Hadoop生态系统示意图

以通过Hadoop客户端或者Pig语言、Hive数据仓库、Oozie作业流等接口系统提交。

MapReduce

MapReduce计算框架最初是Google为了创建网页索引而创造的。MapReduce框架将整个数据集进行划分，在多个计算节点上并行执行，这种分布式计算模式解决了单机处理能力有限的难题。使用MapReduce框架进行数据分析，通常需要三个操作。

■ **数据加载**：如果用数据仓库的命名方式，应该称为抽取、转换、加载（简称ETL）。从数据源获取的数据，必须经过规范化，再导入MapReduce可访问的存储系统。

■ **数据计算**：从存储系统读取数据，进行处理，再将结果输出到存储系统。

■ **数据抽取**：为了使得处理结果可视化，还要将存储系统的结果进行查询和展现。

Hadoop是应用最为广泛的开源MapReduce实现，于2006年由Doug Cutting创建。目前，Hadoop项目和多个子项目一起共同构成了完整的SMAQ开源生态系统。MapReduce的一个重要特征就是扩展性非常好，MapReduce的各个组件，包括InputFormat、OutputFormat、Partitioner、Combiner等都设计得非常模块化，便于开发和使用者进行变换来适应不同需求。

在海量数据生态系统中，计算模型也不仅限于MapReduce一种，有很多MapReduce计算模型的变种，像HPDC2010会议论文介绍的Twister是一种Iterative MapReduce；而微软也有自己的基于DAG执行流的计算模型Dryad，其与Hadoop的对比如表1所示。

表1 Hadoop和Dryad特征对比

特征	Hadoop	Dryad/ DryadLINQ
编程模型 & 语言支持	MapReduce支持Java，其他语言通过Streaming接口支持	基于执行流DAG，DryadLINQ为Dryad提供了LINQ编程API
数据存储	HDFS	共享目录/本地磁盘
中间数据交互	HDFS/通过HTTP协议点到点	文件/TCP管道/共享内存FIFO
调度	数据局部性/机架感知	数据局部性/基于运行时图优化的网络拓扑
失效处理	通过HDFS实现一致性Map和Reduce任务重新执行	顶点重新执行
监控	HDFS存储和Map-Reduce计算监控支持	支持执行图的监控

Storage

MapReduce计算框架的数据输入和结果输出都依赖存储系统。与传统数据库不同，计算输入是非关系型数据。输入数据存放在不同的文件块上，划分给不同的计算节点，以Key-Value的形式供给Map或Reduce方法。存储系统不仅要满足与MapReduce的接口简单及可扩展，而且需要考虑到数据加载和结果查询。

Hadoop文件系统

作为Hadoop的存储系统，HDFS有以下特点。

■ **容错**：假定失败是常态，允许HDFS运行在普通硬件上。

■ **流式访问**：考虑批量处理能力，着眼于高吞吐率而非访问延迟。

■ **可扩展性**：可以扩展到PB级以上的规模，像Yahoo!、Facebook、百度都有超过2000台规模的集群。

■ **可移植性**：Hadoop是Java语言实现的，不受操作系统限制。

■ **不支持写更新**：假定文件只写一次，简化了副本技术，提高了数据吞吐率，当然现在的Hadoop版本也支持append操作。

■ **本地化计算**：在数据所在地执行计算任务，减少网络传输压力，这点对分布式系统的整体性能是非常重要的。

■ **不支持索引**：HDFS提供了一个类似于标准文件系统的接口，只能进行数据存储和访问，不支持为数据建立索引，无法对数据进行简单高效的随机访问。

Hadoop数据库

在Hadoop上支持索引的系统是HBase。类似于Google的BigTable，HBase是用于存储并索引海量数据的列存储数据库，它属于NoSQL数据库范畴，类似于Cassandra和Hypertable。

HBase使用HDFS作为底层存储系统，因此也具有通过大量容错分布式节点来存储大量的数据的能力。与其他列存储数据库类似，HBase也提供基于REST和Thrift的访问API。

由于创建了索引，HBase可以为一些简单的查询提供对内容快速的随机访问。对于复杂的操作，HBase为Hadoop MapReduce提供数据源和输出系统。因此HBase允许系统以数据库的方式与MapReduce进行交互，而不是通过底层的HDFS。

Cassandra和Hypertable都是具有BigTable模式的类似于HBase的列存储数据库。作为Apache的一个项目，Cassandra最初是在Facebook产生的，现在应用于很多大规模Web站点，包括Twitter、Facebook、Reddit和Digg。Hypertable产生于本地搜索公司Zvents，也是另一个开源项目。比较而言，Hypertable和HBase更加类似一些，它可以看作是HBase的兄弟，它们都基于底层的HDFS，主要区别在于HBase使用Java语言开发，功能更加完善一些，而Hypertable使用Boost C++开发，在某些功能上性能更好，不过功能完备性不如HBase。这两个数据库都提供与Hadoop MapReduce交互的接口，允许它们作为Hadoop MapReduce作业的数据源和输出系统。在更高层次上，Cassandra提供与Pig或者Hive查询语言的集成，而Hypertable已与Hive集成。

数据仓库

数据仓库是SMAQ生态系统的一个重要应用领域，其简化了结果数据的报告和分析。Facebook开发的Hive，是一个建立在Hadoop之上的数据仓库框架。类似于HBase，Hive提供一个在HDFS上的基于表的抽象，简化了结构化数据的加载。与HBase不同的是，Hive只运行MapReduce作业进行批量数据分析，即提供类SQL的查询语言来执行MapReduce作业。

NoSQL数据库

目前为止我们提到的存储解决方案都是依赖Hadoop进行MapReduce，还有一些NoSQL数据库为了对存储数据进行并行计算，其本身具有MapReduce支持。与Hadoop系统的多组件SMAQ架构不同，这些系统实现了“Storage, MapReduce and Query”一体化的自包含系统。

Hadoop系统通常面向批量处理分析，而NoSQL系统通常面向实时应用。在NoSQL数据库中，MapReduce只是个附加功能，作为其他查询机制的一个补充而存在。例如，在Riak里，对MapReduce作业通常有一个60秒的超时限制，而一般来说，一个Hadoop作业会执行几分钟或者几小时。

下面几个NoSQL数据库都具有MapReduce功能。

■ CouchDB：分布式数据库，提供了半结构化的文档存储功能。主要特点是提供很强的多副本支持，以及可以进行分布式更新。在CouchDB里，查询是通过使用JavaScript定义MapReduce的Map和Reduce阶段实现的。

■ MongoDB：类似于CouchDB，更注重性能，对于分布式更新、副本、版本的支持相对弱些。MapReduce也是通过JavaScript描述的。

■ Riak：类似于前两者，更关注高可用性，可以使用JavaScript或者Erlang描述Map-Reduce。

这里再多说两句，NoSQL仅仅是一个概念，它的提出是为了解决关系数据库在性能、

扩展性、容错和易用性等方面的不足。根据业务场景选择合适的数据库，以及进行多种数据库的融合运用是当前流行趋势。

与关系数据库的集成

如图1所示，在很多应用中，数据源存储在关系数据库中，比如MySQL或者Oracle，MapReduce通常通过两种方式使用这些数据：使用关系数据库作为输入源（比如社交网络的朋友列表）；将MapReduce结果重新导入关系数据库（比如基于兴趣生成的推荐列表）。

最简单的，通过组合使用SQL导出命令和HDFS操作命令，带分隔符的文本文件可以作为传统关系数据库和Hadoop系统之间的传输格式。当然还存在一些更复杂的工具。Sqoop用于将数据从关系数据库导入到Hadoop系统，由Cloudera开发。Sqoop使用了Java的JDBC数据库API，因此与数据库种类无关，可以导入整表，也可以使用查询命令限制需要导入的数据。Sqoop也提供将MapReduce的结果从HDFS导入关系数据库的功能。因为HDFS是一个文件系统，所以Sqoop需要以分隔符标识的文本为输入，将其转换为相应的SQL命令才能将数据插入到数据库。

对于Hadoop系统，通过使用Cascading API中的cascading.jdbc和 cascading-dbmigrate也能实现类似的功能。

与流式数据源的集成

如图1所示，关系数据库以及流式数据源（例如Web日志、传感器输出）组成了海量数据系统的数据来源。Cloudera的Flume项目就是旨在提供流式数据源与Hadoop之间集成的方便工具。Flume收集来自于集群机器上的数据，将它们不断地导入到HDFS中。Facebook的Scribe也提供类似的功能。

商业性的SMAQ解决方案

一些MPP数据库具有内建的MapReduce功能支持。MPP数据库具有一个由并行运行的独立节点组成的分布式架构。它们的主要功能是数据仓库和分析，可以使用SQL。

- Greenplum：基于开源的PostreSQL DBMS，运行在分布式硬件组成的集群上，MapReduce作为SQL的补充。Greenplum MapReduce允许使用由数据库存储和外部数据源组成的混合数据。MapReduce操作可以使用Perl或者Python函数进行描述。
- nCluster：Aster Data的nCluster数据仓库系统也提供MapReduce支持。SQL-MapReduce技术可以使SQL查询和通过各种语言（C#、C++、Java、R或者Python）定义的MapReduce作业组合在一起。

其他的一些数据仓库解决方案选择提供与Hadoop的连接器，而不是在内部集成MapReduce功能。

- Vertica：是一个提供了Hadoop连接器的列存储数据库。
- Netezza：最近被IBM收购。与Cloudera合作提高了它与Hadoop之间的互操作性。尽管它解决了类似的问题，但实际上它已不在我们的SMAQ模型定义之内，因为它既不开源也不运行在普通硬件上。

尽管可以全部使用开源软件来创建一个基于Hadoop的系统，但集成这样一个系统仍然需要一些努力。Cloudera的目的就是使得Hadoop更能适应用企业化的应用，而且在它的Cloudera Distribution for Hadoop（CDH）中已经提供了一个统一的Hadoop发行版。

Query

直接使用编程语言来定义MapReduce作业的Map和Reduce方法并不是那么简单方便，而且多个开发者存在大量重复工作。为了减少开发成本，提高MapReduce的易用性和系统的执行效率，SMAQ引入查询层来简化MapReduce操作和结果查询。查询层不仅提供用于描述计算的简单接口，而且支持数据存取方法，并优化MapReduce集群上的执行流程。

Sawzall

Sawzall在Google的*Interpreting the Data: Parallel Analysis with Sawzall*论文中提出，它是解释型、过程式的编程语言，该语言把Reducer

抽象成聚合器（Aggregator），通过重用聚合器来减少重复编码。

Cascading

Cascading对Hadoop MapReduce API进行封装，使其更易被Java应用程序使用，它只是为了简化MapReduce集成其他系统的封装层。

Cascading的关键特性是允许开发者将MapReduce作业以流的形式进行组装，因此能很方便地将Hadoop集成到其他系统中。Cascading本身并不提供高级查询语言，由它而衍生出的一个叫Cascalog的开源项目完成了这项工作。Cascalog通过使用Clojure JVM语言实现了一个类似于Datalog的查询语言。尽管很强大，Cascalog仍然只是一个小范围内使用的语言，因为它既不像Hive那样提供类SQL查询，也不像Pig那样是过程式的。

Pig

Pig由Yahoo!开发，是Hadoop的子项目，该项目设计了一套日志分析语言Pig Latin。Pig Latin语法源于SQL，提供了与SQL中where、groupby、orderby等类似的子句。Pig目前已集成在Cassandra和HBase数据库中。

Pig允许开发者通过UDF（User Defined Functions）定制所需功能。这些UDF使用Java语言书写。尽管比MapReduce API更容易理解和使用，但Pig要求用户去学习一门新的语言。某些程度上它与SQL有些类似，但又与SQL存在不同，因为那些熟悉SQL的人们很难将SQL知识在这里重用。

Hive

如前所述，Hive是建立在Hadoop之上的开源数据仓库，它提供了一个非常类似于SQL的查询语言，而且提供一个支持简单内建查询的Web接口，因此很适合于那些熟悉SQL的非编程用户。

与Pig和Cascading需要编译相比，Hive提供了即时查询。这点对于那些已经成熟的商用系统来说更加自然，因为它提供了一个对非技术用户的更加友好的接口。Cloudera的Hadoop发行版里集成了Hive，而且通过HUE项目提供了一个更高级的用户接口，使得用户可以提交查询并且监控MapReduce作业的执行。

Solr

数据查询和数据摘要是大规模数据处理系统的重要组件。Hadoop数据库例如HBase提供了对数据的简单查询，但并不具备复杂搜索的能力。为了解决复杂搜索问题，开源搜索和索引平台Solr常常与NoSQL数据库组合使用。Solr使用Luence搜索技术提供独立的搜索服务器产品。例如，考虑一个社交网络数据库，MapReduce可以使用一些合理的参数用来计算个人的影响力，这个数值会被写回到数据库。之后使用Solr进行索引，就允许在这个社交网络上进行一些操作，比如找到最有影响力的人。

最初在CENT上开发，现在作为Apache项目的Solr，已经从单一的文本搜索引擎演化为支持导航和结果聚类。此外，Solr还可以管理存储在分布式服务器上的海量数据，这使得它成为在海量数据上进行搜索的理想解决方案，以及构建商业智能系统的重要组件。

总结

MapReduce、Hadoop及其开源生态圈的形成，使得在普通集群上搭建海量分布式计算成为现实，这大大降低了硬件成本。分布式计算、分布式存储以及用户友好的查询机制，所形成的SMAQ架构使得海量数据处理系统可以通过小型团队搭建。当然这些变化也降低了海量数据分析和数据仓库的门槛，使得这个领域的产品变得低廉，促使这个领域产生新的产品、服务和组成方式，开启了廉价数据驱动产业的新纪元。

杨栋

百度分布式高级研发工程师，从事Hypertable、Hadoop及流式计算的研究和开发。

责任编辑：董世晓（dongsx@csdn.net）

与开发者一起走向世界

华为云应用集成部总监刘成专访

记者 / 陈博

华为手机凭借其强大的全球化的优势，对中国开发者产生了巨大吸引力。“智汇云”平台的建立及“智汇云”开发者大赛的举办是华为深入智能手机市场的重要举措。为了让开发者对华为“智汇云”战略有更深入的了解，记者采访了华为云应用集成部总监刘成。

“智汇云”：开发者与用户间的桥梁

《程序员》：华为建设“智汇云”的背景是什么？您主要负责哪些方面的工作？

刘成：“智汇云”是一个面向全球同步发布、以精品应用为主的全球化网络平台。华为建设智汇云的初衷，不仅是为了更好地销售华为终端，更希望能以华为终端为载体，为用户提供更多的增值服务，解决目前在国内以及部分国家，因缺乏应用商店，或有些应用商店非官方渠道所产生的问题。

“智汇云”平台是由我负责搭建的，包括技术平台的建设、应用的合作以及开发者社区的建设，都是由我们团队完成的。

《程序员》：2010年，华为首次提出了“端管云”战略，那么“智汇云”在整个战略体系下扮演着怎样的角色？

刘成：端、管、云的实质是终端、网络、应用的概念。端，就是我们和用户的界面，需要做到终端的智能化、信息的多媒体呈现；管，需要的是支撑大流量，做到智能化的管道；云可以理解成把业务IT化，最主要的就是新一代数据中心和新一代业务平台。“智汇云”作为“端管云”战略的一部分，不仅是用户下载应用最终的出口，同时也是华为终端与广大开发者合作的平台。在这个平台中我们搭建了开发者社区，为开发者提供很多的帮助。我们与开发者合作的所有应用，用户都可以从“智汇云”上下载。现在我们拥有200万用户，每天的下载量达十几万次。

以三大特色应对全球竞争

《程序员》：目前，以软件应用推动硬件销售的模式取得了巨大成功，但竞争也十分激烈，与其他厂商相比，华为有哪些特色？

刘成：目前，市面上所有的华为手机都预置了“智汇云”应用，它具备三个特点。

第一个是“精品”。“智汇云”区别于谷Google的Android Market，虽然Google的Android Market有40多万应用，但整个Android Market鱼龙混杂。华为会为用户精选每一款应用，例如，在Google的Android Market中可能有很多款不同的Facebook应用，但在“智汇云”内，我们会为用户精选一款，这样极大地方便了用户。

第二个是“正版”。“智汇云”内的应用区别于国内的山寨产品以及来自第三方市场的应用，为用户解决了版权和续费等问题，这也是我们的优势。

第三个是“免费”。“智汇云”上应用的下载是完全免费的，相当于为终端用户提供了增值服务。华为会依据应用的质量和市场状况向开发者支付费用，并为开发者提供其他一些诸如广告、道具等盈利模式。

《程序员》：华为通过哪些措施来保证“智汇云”应用的精品化？

刘成：首先，在应用的来源上，我们会优选合作伙伴。我们着重考察应用的定位，从UI界面等方面进行评估。

其次，我们的审核更加严格，我们有一个50

人的审核团队专门负责应用的测试、评估。必须保证应用与终端的适配性，这样才能拥有更好的用户体验。在Android Market上跨机型的适配存在很多问题，而“智汇云”上的应用只面向华为终端，所以其适配性更强。

《程序员》：与iOS不同，Android平台存在不同厂商间的竞争，华为将如何在竞争中脱颖而出？

刘成：首先，我们与Google有着十分紧密的合作关系，早在2008年我们就进入了OSA，因此比其他厂商能更早地在新平台上进行产品的发布。例如我们推出的MediaPad就是全球首款使用全新Android 3.2系统的平板电脑。

其次，我们在平台上做了一些针对区域化市场、本土化市场的改进。例如在内地，我们在输入法、文字的处理等方面，做了很多拥有自身特色的调整。

《程序员》：“智汇云”开发者大赛的参赛者基本上都来自中国本土，华为如何将这种“本地化”特色转化为“全球化”竞争的优势？

刘成：在移动互联网的行业，中国的产品、从业人员、开发群体仅次于美国，中国开发者拥有巨大的潜力。而海外很多地区和国家的产业链不够健全，存在更大的机会，所以我们希望通过“智汇云”平台与国内的移动开发者一起走向海外。同时这样的合作，也是华为提升自己，并增强“智汇云”平台能力和应用的方式。

《程序员》：8月3日，华为推出了云服务平台，为用户提供了无线推送服务，这项服务有何特点？

刘成：无线推送是华为“端管云”战略里“端”和“管”强强结合的结果。其中的核心技术就是无线IP的推送能力，这是华为的传统优势。华为在对整个移动网络的理解上有很多积累，我们利用了公司内部的资源，把最关键的核心技术做好。从目前的测试结果来看，我们的功耗较低并且成功率远高于其他平台。

多种合作方式帮助开发者成长

《程序员》：华为与开发者合作的模式是怎样的？

刘成：我们与开发者有五种合作模式。

第一，应用商店，这是最快捷、最广泛的合作模式。

第二，对于受用户欢迎、下载量大的应用，我们会提供SD卡预置及手机预置的方式，使其能更容易的让用户体验。

第三，通过API的方式，做到功能的深度集成。它是一种深度合作的方案，比如去年我们发布的HiQQ手机，以及与新浪、高德的合作，这不仅仅是预置客户端，而是双方在内容、应用、功能上深度的合作。

第四，针对优秀的应用，我们会发挥华为在全球的销售优势，把优秀的开发者及解决方案呈献给华为的客户。

第五，我们称为开发者的经济服务模式。包括软件外包、功能外包以及完整的应用开发外包，这是我们与开发者合作最深度的层次。

这些合作最重要的目的就是帮助开发者盈利、发展，使其能推出更多的精品应用。

《程序员》：您认为目前中国的开发者面临哪些方面的问题？您对开发者有哪些建议？

刘成：主要存在两方面的问题。

第一方面是目前很多应用的功能非常相似，应用的开发缺乏明确的定位：应用针对哪些用户？服务于哪些需求？如何找到更多创新的思路？这是整个产业链遇到的问题。所以我们希望与合作伙伴一起，进行深度的探讨，做出更多差异化的产品。

第二方面是目前很多开发者遇到的收入问题，我刚刚提到华为与开发者合作的五种模式，就是为了能更好地帮助合作伙伴，让开发者有收入、能成长。

我认为开发者应从三个大的方向入手：

首先是面向各企业各行业的应用，例如物流、金融保险等行业，这些行业有很大用户群体，需要IT化，更需要移动化，它们需要更好的行业服务，我认为这个领域是蓝海市场。

其次是手机网游，是很好的让开发者摆脱盗版，摆脱无收入的模式。

最后是本地生活化的应用需求，例如各类服务性行业都需要通过移动客户端进行推广、发展用户、服务用户。

成都优聚：bada更具盈利前景

记者 / 杨鹏飞

这是一个成功的创业故事：优聚（GoodTeam）是一家成立不久的公司，涉足bada操作系统平台开发，并获得了2010年全球bada应用软件开发者挑战赛的冠军。优聚总经理李万鹏告诉记者，这是一片蓝海，等待冲浪者们去挖掘新的宝藏。

结缘bada，开启蓝海之旅

《程序员》：你们是如何知道三星bada平台的？对bada平台有怎样的感受？

李万鹏：2010年5月时，三星举办过一次bada开发者大会，我们就去参加了，当时是第一次听说三星有自己的手机操作系统平台，觉得挺有兴趣，从此就跟bada结缘了。

对bada的第一感觉是很像iOS，两者都是封闭的平台。虽然Android正在高速成长，但没有人怀疑iOS才是最赚钱的平台。bada是三星自己的手机操作系统平台，基于三星强大的手机销售量，我们预感在bada上开发游戏会获得很多机会，所以从5月开始制作bada平台的游戏。

《程序员》：你们开发的游戏获得了哪些奖项？比赛过程有何感受？

李万鹏：到目前为止，优聚共开发了30多款游戏，以其中的2款为主打，分别是《钢丝英雄》和《帝国塔防》。《钢丝英雄》于2009年获得中国移动最具创新奖，同时也被美国媒体评为“年度最佳Android游戏”，登上 Top 50榜，而《帝国塔防》则获得2010年三星bada中国开发者大赛年度大奖。

我们是抱着学习和交流的目的去参赛的，对能否获奖没有太多的思想包袱。当然既然参赛了，还是希望把参赛作品做好，毕竟是一个向大家展示自己的机会。获奖出乎意料，令我们非常激动。

游戏《帝国塔防》

坚持梦想，做“简悦”游戏

《程序员》：能介绍一下你们的团队吗？

李万鹏：团队的创始人都是来自于同一家公司，之前从事嵌入式游戏开发，重点在芯片上的游戏设计、开发、维护。2008年8月Google发布第一款Android手机G1时，创始团队认为移动互联网的机会来临，所以选择了Android的游戏开发。开始还主要利用业余时间凭兴趣研究开发。后来大家认为只有独立创业才有机会，所以2009年4月成立优聚软件，同年3月我们发布了第一款游戏，开始了漫漫的创业之路。从被大家不理解、不认可的状态到目前市场的火热，我们一直坚持产品第一，把做好游戏献给用户定为我们的第一目标。

目前团队共有26人，我们把自己定位为游戏公司，我们的一切工作都会以游戏开发为重心，下一步的规划，还是坚持最初的梦想，做“简悦”的游戏，让游戏简单化的同时给大家带来快乐。

《程序员》：在你们下一步的发展中有融资计划吗？

李万鹏：一直以来都有VC来找我们，不过目前暂时没有融资计划。因为我们一直认为自己还是一个很小的创业团队，还有很多路需要踏实地走过去，包括游戏开发、团队管理、盈利模式等。我们希望能够通过今年的打拼，建立良好的基础，为团队的发展做好准备。

bada更具盈利优势

《程序员》：bada平台和Android平台相比，哪个更容易盈利呢？

李万鹏：bada平台的整体盈利状况不错，当然在所有平台的盈利都取决于产品的质量，这样玩家和用户才愿意向你埋单。目前看bada的盈利要胜于Android，因为封闭平台的支付模式相对更加快捷。

《程序员》：你认为bada的优势在哪？你更侧重于哪个平台？

李万鹏：bada的优势在于三星有强大的硬件生产线和销售体系，bada的终端占有率已经超过4%，由于三星终端销量的不断增长，盈利也在稳步上升。同时bada也是一个封闭的平台，有方便的应用商店和清晰的盈利模式，开发者不必为盈利犯愁。

优聚2009年成立时，一开始我们专注于Android的游戏开发，从2010年5月开始进军bada平台。目前所有的游戏都集中在Android和bada平台，Android游戏共有30多款，bada游戏有20款左右。

高品质决定成败

《程序员》：你们觉得bada平台的开发门槛高吗？跨平台开发有哪些困难？

李万鹏：门槛的高与低主要看跟哪个平台对比。与Android平台相比，bada的门槛要高一些，毕竟Android还可以使用Java开发，而目前国内Java的普及率非常高。但对于有C或者C++基础的开发者来说，开发bada应用的门槛也不算高。毕竟游戏设计中不管你是用J2ME、Java还是C++，实现原理都是一样的。今年我们配合三星在校园开展bada游戏培训和制作，同学们在短短的3个月就能完成一款2D的bada游戏，说明bada是易于上手的。

跨平台最大的困难是设计手机底层的功能，包括音乐音效播放、震动、摄像头、屏幕绘制方式、联网操作等。而游戏的算法和逻辑基本上都是一致的，iOS或Android也都可以用C++编写。所以，移植的重点在于开发过程中需注意手机本身特性。

《程序员》：bada是个新兴的平台，可以为开发者分享一些你们的成功经验吗？

李万鹏：bada平台并不神秘，只是它有自己的规范、接口和适用的语言。开发者需要对bada的基础API进行必要的了解，对游戏来说，大部分代码其实都无须重写，只需要针对bada平台，做一些特殊的处理就可以。

与Android平台相比，bada的应用提交和审核都更加严格和细致，三星对于游戏的QA（Quality Assurance，品质保证）非常的严格，有利于树立bada品牌的优质形象，让用户放心地使用软件或游戏。对于开发者而言，能做出顺利通过审核的bada应用更能彰显自己的技术和实力。

《程序员》：作为开发人员，你们觉得三星在哪些地方还需要进一步完善？

李万鹏：我的建议是：

第一， 三星在线应用商店的应用提交审核周期如果能缩短，将能够使开发者更快地更新产品，提升用户体验；

第二，国内访问三星开发者后台时，速度慢，导致需要反复几次提交软件；

第三，bada用户抱怨有时无法在三星在线应用商店正常下载；

第四，希望增加更多的用户支付方式，让支付更加简便；

第五， 希望增加游戏内支付，让游戏的盈利模式多样化。

手机社交游戏设计中交互理念的渗透

文 / 郑金条

延长玩家与游戏的生命周期

我从游戏设计中了解最多的除了游戏本身的机制效能，就是游戏化概念对其他产品价值及其衍生属性的影响。从本质上讲，游戏所提供的就是虚拟性的服务，而这层服务从用户获得体验那刻起开始生效，不仅仅是单纯为了一个漂亮的留存数据，更在于这种以时间投入为置换准则的泛娱乐形式能否起到愉悦用户的目的。而付费性一直是一个附属品，一个认可游戏价值的玩家终究能够自然发展为付费玩家，并且他们乐意在投入精力的同时帮助游戏探索和发展，延长玩家和游戏双重层面的生命周期，而这两者生命周期的延长就可能意味着一款游戏在用户营销中的成功。

这在任何的产品层面几乎都是适用的，而社交游戏对用户设想的市场捕捉和可用性、易用性反馈以及依托它们对用户认知的角度源源不断的服务更新。对于暂时性偏单机化或者交互性不强的手机游戏而言，相信用户的选择仍然是一条保证开发者不致于完全迷失在纯粹创想迷局中的准则。尽管类似Zynga这种拥有超级用户数据分析能力并真实依赖数据分析来判断用户喜好从而不断进行修正的公司还不算太多，但很重要的一个层面在于玩家在游戏进程中所处的量级并不一样，开发者利用了玩家的时间差，以少部分领先者为体验对象，再根据他们的反馈修正并服务于绝大部分的玩家。

把游戏变成一种服务

相对来讲，手机游戏的尝试性反馈修正并不能如社交游戏那样灵活，甚至可能缺乏有效的受众反馈意愿，只能更多选择基于开发者对市场的判断，但同上文所提及的一样，如果把游戏变成一种服务，那么其开发的立意角度就完全改变，用户将不再是一个更新的被动接受者而是一个被考量的需求、体验被尊重的客户。

手机游戏在用户选择性增多的情况下，一再遭受只被短暂激活而不被留存的困扰，问题在于对用户而言这类的App可有可无的成分太高了，相对来讲被尊重的需求将从用户感受的角度来激活他们的选择性偏好。

从这个层面上看，游戏化的概念主要在于用户对产品本身的介入深度和黏着属性。早先我们尝试就玩家的游戏动机和类型属性进行研究，但后来逐步领悟到玩家只是投入等值时间并期待从游戏中置换娱乐的行为，而付费行为只是玩家在游戏娱乐中水到渠成的结果。

让游戏成为用户的精神载体

在我们看来，游戏是一种进程当刻的精神愉悦（当然也有非进程当刻的牵挂），或者从文化层面上来说，游戏是另外一种形式的阅读体验载体。我们知道不同文化载体都在传输理念，而游戏的脚本范畴也在复制相似的层面。对于玩家而言，虚拟世界的营造和挣扎与他们在现实生活中所遇到的情形相似，完全可以匹敌于影像和文本载体的传播力。或者说，游戏和书刊以及影像都是人在满足生存之余，开拓闲置时间的一些精神性选择。

在明白这个层面后，就能更清晰地明白为什

么同样作为文化载体，书刊在用户（至少现在是这样）的选择性机率上远远小于游戏，原因可能在于游戏加深了玩家在体验时的介入深度和互动属性。因此无论是出于何种意愿，我们似乎能阐释用户更乐意于成为玩家的这层可能性，特别是在社交游戏层面，玩家的被服务意识前所未有地被加深了，不会在他们购买了产品之后脱离生产和供销的链接。对于传统手机游戏而言，这种脱节的存在也相当明显，但在更多的社交游戏公司介入手机游戏研发或者更多公司转向手机社交游戏后，开启了全新的用户认知。

正如前文中将游戏还原为一种虚拟化形态的服务（事实上正是服务的这种属性使开发者更关注玩家在游戏中的状态），对用户而言，产品的使用性能和持久的服务是购买的首要选择，并且在游戏进程中，玩家在生活中的素养会慢慢在游戏中体现出来，甚至可以诱发出某些在现实中被隐藏得很深的行为。因此在研习玩家的习性和心理时，还需要关注一些非显性的问题。这并不是挖掘人性的阴暗面，而是释放玩家在现实生活中被压抑的一些情绪，就像看一部感人的电影可以让人肆无忌惮地哭泣，同样一部糅合玩家情绪的游戏也可以让玩家在游戏中得到情绪的舒缓。因此，短期尝试并不能完全检测游戏的效能和玩家的忠诚度问题，所有的症结都需要回归到玩家乐意长期投入游戏进程，不管是游戏时发自内心的喜欢还是过后不时的牵挂都有相应的聚合力。此时，所有的环节都明确地指向了玩家留存游戏的可能性。

说起留存，手机游戏已经慢慢从以下载基数为表征的概念范畴延伸到以下载用户的活跃度为评判标准，这种趋势在免费游戏慢慢盛行后将明显，不管是从广告展示还是IAP角度，活跃用户才是两者的有效驱动。开发者单方面提供了服务内容，并且期待这种单向服务能够和玩家此刻的需求有所吻合，让事态由单方面给予演变成为双方的长期互动。事实上，这是开发者以游戏机制和价值所发起的玩家留存保卫战，似乎这并不是博弈进程，玩家在整体的游戏空间中有多重选择（其他游戏），甚至还有不进行游戏的选择，以及只进行到什么程度的选择，而开发者最大的说服力就是让玩家在每一阶段都能收获心情价值或者能够和自己的游戏玩家在交互中感受到虚拟的真实性。

这并不是挖掘人性的阴暗面，而是释放玩家在现实生活中被压抑的一些情绪，就像看一部感人的电影可以让人肆无忌惮的哭泣，同样一部糅合玩家情绪的游戏也可以让玩家在游戏中得到情绪的舒缓。

成功之匙：提升用户间的交互性体验

到目前为止，主流化的社交游戏设计中用户间的交互依然是最核心的环节，在手机游戏层面，无论是从单机版游戏的成就系统比照（数值攀比性）还是交互型游戏玩家之间的互动都有类似社交渗透的趋势，慢慢进行游戏理念的转移。对于社交游戏而言，好友生态链推动了玩家的游戏持续力，关键在于交互发挥了价值，其产生的隐性受迫力和牵制力一定程度上左右了玩家对游戏的持续时间投入，因为游戏本身因为交互可能已经完全演变为一种类似于使用游戏语言进行即时聊天的独特交互方式。这种糅合情感元素驱动玩家协作以带来更具黏性价值维度的方式正在慢慢向手机游戏领域渗透。

情感黏着力可能是玩家除了游戏本身之外值得继续投入的另外一层动力，如果缺乏情感因素的左右，那么游戏在玩家面临多重选项的情况下可能时刻面临着流失的问题，并且这种流失将是随意性的。没有情感因素所支持的游戏牵制力，就没有办法让玩家达到和游戏任何一个微小进程都休戚相关的程度，漠视是一种很可怕的行为，一旦玩家和游戏之间撇清了价值关联度，流失都将是自然而然的。

还需要明确社交游戏的成功并非只是Facebook形式的成功，更在于它本身的娱乐性和用户间交互性的提升，社交游戏本身对用户的吸引力促成了大量定期访问的玩家，活跃了整体的社交氛围并影响了用户之间的交互方式，从信息文本衍生为游戏行为。因此这种属性在不同平台之间的迁移并不是困难的事情，它的基础架构不在于平台类型而在于平台的用户聚合能力，有聚合能力的平台就能为玩家之间的交互提供无限可

能，可以更形象地外化为一款有高下载量和高活跃度的手机社交游戏，其自身为用户间的交互所提供的空间和Facebook所能提供的属性则完全是相似的。

正因为这种交互特性的存在，使得即使社交游戏病毒式传播的属性不能完全在手机游戏领域发挥效能，但我们也能看到类似Heyzap这种类似Facebook游戏推荐功能正在手机App之间慢慢推出。而同样手机社交游戏的这种交互性需求让玩家在App的下载端开始自发地邀约好友，开始彼此之间的交互体验。

精雕细琢方成大器

我记得Neil Young（手机游戏公司ngmoco，隶属于DeNA）在一次演讲中指出我们已经进入了一个要像做电影一样做游戏的时代，任何一款游戏都需要各方携手精雕细琢，站在玩家的角度尽可能做多方位的思考，而这也类似于我们全文所着力论述的从游戏下载安装起才是开发者真正开始服务的论调。因为只有当玩家将游戏视为生活环节的一部分时，才会乐于为营造更好的游戏生活而进行游戏消费。事实上，这种融入感可以呈现为游戏环境的营造让玩家觉得他们拥有自己所属的虚拟环境；能否吸引玩家真切地关注游戏中的每一个环节，让玩家感受到这种变化与自己休戚相关。

如何让玩家重复回到游戏中是所有开发者必须要面对的问题。是什么因素促成了玩家下一次的登录？是因为游戏值得玩家一再付出时间还是因为游戏有了新的发展、好友有了新的动态？开发者如何在玩家自觉和被动的回访中找到更好的平衡点？ 上文我们提到了玩家的情感因素，但再深的情感也需要转化成具体的操作，玩家需要的是具体的执行牵挂。终归来讲，让玩家为解决某些问题而回到游戏中的方式更具可操作性，让他们清晰明白做什么才能解决游戏中遇到的障碍，让他们真切地感受到他们的每个操作都能够对整体的游戏进程产生影响，而玩家本身和游戏执行程度休戚相关。

郑金条

游戏邦负责人，游戏邦主要关注和解析国内外社交游戏和手机游戏领域，并定期做深度行业阐述。

“建立企业中的用户体验力量”系列之一

当今时代，包括苹果、谷歌、Instagram等不同类型、不同规模、不同领域的企业和团队的成功用户体验案例，时时激发着你的热情、激励着你的梦想，向你描绘了运用用户体验力量的前景。然而，一旦开始尝试在企业中构建用户体验的力量时，你会发现：用户体验的投入与回报难于衡量，用户体验的工作方式可能与团队以往的工作方式难于融合，用户体验人才难于获得、难于领导、难于培养，用户体验的力量难于持续成长……本系列文章，将针对以上问题，为您分析如何在企业中构建用户体验的力量。

用户体验的价值

文 / 吴卓浩

苹果凭借不断创新的用户体验重塑了世界上的多个行业；谷歌秉承“以用户为中心，其他一切水到渠成”的理念改变着人们的生活方式；新浪微博在用户体验上借鉴创新，成为炙手可热的产品；Instagram、Kik、Quora、Tumblr这些由只有几个到几十个员工的小企业开发出来的创新体验产品却能震动世界；甚至海底捞也向人们展示了关注用户体验可以为传统行业带来怎样的改变。

用户体验工作面对挑战

都说用户体验重要，但是究竟有多重要、用户体验的价值究竟如何衡量，却是一个无法回避、又很难回答的问题。有人从定性的角度举例说苹果的iPod、iPhone，初期在技术、成本等条件和竞争对手基本相等，甚至品牌在所竞争的领域还略有劣势的情况下，能够压倒对手，靠的就是用户体验。而苹果在用户体验上取得优势后反过来又能在成本、技术上压倒对手。

也有人试图将用户体验的价值量化，比如互联网产品中常用的A/B测试法，控制绝大部分条件不变、只改变少量用户体验相关的条件，对比产品的相同指标在修改前后的数据变化：用户完成一个任务所需要的操作步骤/时间是否减少了？用户完成一个任务的成功率/效率是否提高了？用户使用产品的时间/频率是否增加了？用户转化率/留存率是否增长了？用户操作出错率/投诉率是否降低了？还有人通过建立复杂的数学模型，来分析量化产品全局中，各个用户体验贡献点的具体价值……定性好定，定量却难量。因为量化衡量要求被测对象能够被直接或间接分离出来，与参考值相比较，从而得到被测量的大小。然而，你无法真正控制一个产品的研发过程中所有的条件都不变，分离出用户体验工作、去量化其带来的价值。

要理解和认同用户体验的价值，困难主要在于存在两个误区：第一，试图把用户体验工作的贡献从全部工作的贡献中分离出来；第

实现良好的用户体验需要各成员协同合作

二，把“用户体验的价值”等同于“用户体验人员的价值”。

用户体验工作的内容

在今天，用户体验已经成为和产品研发、运营全程密不可分的一部分。通常来说，在产品前期，用户体验研究员、设计师和产品经理一起收集分析用户需求、策划和检验产品概念。在产品中期，用户体验设计师和产品经理、核心研发工程师一起探索、检验，确定产品设计（包括交互设计和视觉设计）。在产品后期，用户体验设计师、界面工程师和研发工程师一起实施、优化产品，用户体验研究员配合进行用户测试、可用性评估。产品发布以后，用户体验研究员、设计师和产品经理、研发工程师一起跟踪改进产品。视觉设计则往往根据产品的不同，在不同的时间点进入：有的在产品前期就要开始参与产品视觉和品牌的策划和设计，有的在产品后期才参与产品实施优化。其中，在产品前期和中期的用户体验工作最为重要，因为这时的探索和验证能极大程度地减少产品研发过程中可能出现的错误和风险，从而从全局上优化研发过程。同时，随着产品与社交元素相结合的兴起，用户体验设计和研究协助产品经理、运营人员、市场人员在产品设计之中就融入产品运营、营销元素，让产品、运营、营销之间相互促进，也正在成为趋势。

而且，在实际工作中往往各种职能的界限并不会被严格的划分，也不会在每个项目上都完整的配备上面提到的每一个角色。同时，在实际的工作中也并不存在所谓的标准人员配备和分工，比如有些人认为产品经理应该是产品设计的灵魂和主导者，但是在实际工作中既有资深产品经理同资浅设计师合作的情况，也有资浅产品经理同资深设计师合作的情况；既有工程师只擅长编写代码的情况，也有工程师对于产品全局架构很有想法的情况。

产品经理、用户体验人员、工程师、运营人员、市场人员，既需要从各自不同的角度对产品提出要求，也需要了解和自己相配合人员的要求、产品全局的要求，只有这样才能避免无谓的误解和矛盾，更有效地构建产品。用户关心的是最终抵达他们的用户体验综合呈现的效果，而不是哪些人在哪个产品环节上做出了一个好的用户体验。这涉及产品设计和实施的各个环节，虽然用户体验工作是用户体验人员的全职工作，但是如果产品经理为了平衡项目各方而没有在最关键的用户体验点上坚持、工程师为了技术上比较容易实施而没有按照产品设计执行、运营和市场人员为了运营推广效果而伤害用户体验，任何一个环节上出现的问题都会导致产品在用户体验上的失控。整个团队只有齐心协力、每个人为达到最终的用户体验效果而努力，才可能真正创造出好的产品。所以说，用户体验的工作是完全融入产品研发与运营的全程中的，并且需要团队的全体成员共同参与。

实现用户体验人员的价值

用户体验工作的贡献无法从全部工作的贡献中单独分离出来，用户体验的价值并不等同

用户体验人员的价值。正是因为用户体验与产品研发、运营全程密不可分，使得无法区分出哪部分是工程师的贡献、哪部分是产品经理的贡献、哪部分是用户体验人员的贡献。从定性的角度分析，每方面的贡献都很重要；从定量的角度分析，却无法计算每方面贡献所占的比重。其实，很多职能之间都有类似的争执，而且历史更加悠久，比如生产和销售、产品和运营。不过，争执归争执，谁又能真正分出高下呢？与其纠结于非要量化分出谁的价值具体有多大，不如首先认可用户体验的价值，最大化的发挥出用户体验与其他各种职能相互激发的能量。

另外，我们看到今天大部分成功的高科技企业都是由工程师所创办，而由用户体验人员创办或者联合创办的则凤毛麟角。从某种意义上来说，这是用户体验的价值不能在更大范围内受到认同的一个非常重要的原因。

工程师具有编写程序实施自己想法的能力，这是一个很大的先天优势，因为即便最初创造出来的产品不好用，但是只要有人用，就会有机会不断得到改进，最终成为一个成功的产品。虽然用户体验人员有很棒的想法，但是自己不会编写程序，就只能借助于程序员的力量才能把想法实现。

所以在现实中，用户体验人员的角色更多的定位在“帮助他人、共同成功”的角色，利用自身对用户的关注和洞察，以及在创造性的思维和表达上的优势，帮助团队更加高效的研发产品，让团队少走弯路、更容易获得成功。

但同时，也看到越来越多的用户体验人员通过提升自身的综合素质，已经或者正在成为优秀产品、优秀研发团队的创造者和推动者，比如在国际上见到越来越多设计师创建的成功公司；在创新工场见到越来越多用户体验人员作为创始人的创业团队；在越来越多公司中见到用户体验/设计副总裁。

在综合能力上取得突破后，他们在用户体验思维和方法上的独特优势得以淋漓尽致地发挥出来，在产品创新、团队建设中都走出了与众不同的道路。他们取得的成就不仅仅属于他们自己，更为其他人树立起榜样和信心，成为推动用户体验事业的重要力量。

细致入微的实施用户体验工作

用户体验需要自上而下的推动，这不是一种态度，更需要施行具体的工作：

1.改变企业的组织、建立用户体验团队、吸引和培养用户体验人才、让用户体验人员在企业决策层拥有发言权；

2.改变企业的流程，让用户体验工作成为计划的一部分，并且评估用户体验工作的效果；

3.改变企业的环境，鼓励创造性思维和探索，让团队协作更多、更顺畅的发展；

4.改变企业的激励机制，让用户体验的员工获得与工程师员工同等的待遇，让跨团队工作的用户体验员工能够获得公平的团队奖励；

5.改变企业的氛围，让每个人都为最终的用户体验考虑，培养他们正确的用户体验思维方法。这些改变有的容易，有的则很难。

但是相比做出改变可能获得的巨大收益、以及不做改变将遇到越来越强的挑战，这些改变不仅值得，而且是必须的。同时，用户体验人员也需要更积极主动的提升自己的综合能力，才能获得更大的影响力，为产品研发作出更大的贡献。

对于用户体验的认同就像是一种信仰。只有开发者真正的信仰它，设身处地的从用户角度出发时，才能得到它带来的好处；如果开发者没有真正的把它作为产品开发的信仰，那么会无法理解它，更无法获得它带来的收益。如果不能把用户体验植入企业的核心价值和文化，用户体验就只能作为可有可无的锦上添花。

你会选择这个将给你带来巨大改变和巨大收益的信仰吗？

吴卓浩

创新工场用户体验总监，曾创建谷歌在中国的用户体验团队。

Kinect引领人机交互变革

文 / 李斌，吴国斌

Kinect凭借其出色的体感互动能力，打出“You are the controller！”的口号，正在引领着一场人机交互的变革。

什么是Kinect？

Kinect是一个Xbox 360外接的3D体感摄影机（如图1），利用即时动态捕捉、影像辨识、麦克风输入、语音辨识等功能让玩家摆脱传统游戏手柄的束缚，通过自己的肢体控制游戏。同类产品有任天堂Wii、Play Station Move，但它们必须让玩家手里拿一个或者多个设备，才能完成所谓体感互动。

图1 备受瞩目的Kinect

Kinect最早定位是XBox 360外设，不需要任何道具即可完成整个动作识别和捕捉，使用了由PrimeSense提供的Range Camera技术。Kinect还可以进行3D立体语声识别。

Kinect的主要识别算法和软件部分都是由微软旗下的一个游戏工作室提供。国内外也有一些所谓可见光或者红外识别公司，也是从这家公司拿到一些著作权，不过它们的产品在精度上跟微软的Kinect还有很大差别。

Kinect的历史

2009年6月1日Kinect在E3上首次公布，当时的代号是“Project Natal”，遵循微软以城市名作为开发代号的传统。在E3 2009上，Kinect的骨骼捕捉技术已经可以在30Hz的条件下同时捕捉4个人的48个骨骼动作。

2010年3月25日，微软宣布将在E3 2010期间召开名为“初生计划全球首秀”的发布会上公布Kinect的发售日期。这个发布会于2010年6月13日晚在Galen Center举行，在会上微软宣布“Project Natal”正式命名为Kinect，取意为“kinetic”（运动）和“connect”（沟通）的融合。微软在这次发布会上同时宣布，Kinect将于2010年11月4日在北美正式发售。

Kinect for Windows SDK 简介

目前的Kinect for Windows SDK Beta版来自微软研究院，发布于2011年6月17日，是一个非商业授权许可，商业授权将在下一个版本中提供。另外Kinect SDK目前只支持Windows 7操作系统，开发环境使用Visual Studio 2010 Express或以上版本。Kinect SDK支持的开发语言有C++、C#和VB.NET。

下面介绍一下Kinect的整体结构，如图2所示，Kinect一共有三个摄像头，中间的是RGB摄像头，用来获取640X480的彩色图像，每秒钟最

多获取30帧图像；两侧是两个景深（3D Depth）传感器，用来检测玩家的相对位置，原理和人眼立体成像是一样的，不过这两个传感器使用的是红外线。Kinect两侧是麦克风，主要用于语音识别。下边还有一个可移动底座，用来调整Kinect的仰角。

图2　Kinect整体结构

Kinect主要包括以下几个功能。

1. 骨骼追踪，根据在Kinect视野范围内移动的一个或两个人的图像进行骨骼追踪，可以追踪到人体上的20个节点。

2. XYZ——深度摄像头，获取标准彩色图像流附加深度数据，来表示物体与Kinect传感器的距离。它的原理是利用双眼的视觉差，能够判断我们面前某一个物体基于我们当前位置的距离，基本上都是通过红外方式来做判断。由于人体散发出来的红外是一致的，因此无论是什么人种，都不会受到干扰。

3. 音频处理，与Microsoft.Speech的语音识别API集成，使用一个具有消除噪音和回波的四元麦克风组，能够把声源附近有效范围之内的各种信息捕捉到。

那么如何使用Kinect进行开发呢？Kinect for Windows SDK Beta提供了非常尖端和复杂的软件库和工具，帮助开发者充分利用基于Kinect的自然输入、信息捕获以及对真实世界事件的反应等特性进行开发工作。Kinect传感器通过该软件库与应用程序进行交互，如图3所示。

Kinect for Windows SDK Beta包含两个重要的API，分别是NUI API和Audio API。下面详细介绍一下它们都提供哪些接口，以及利用这些接口可以捕获和处理哪些数据。

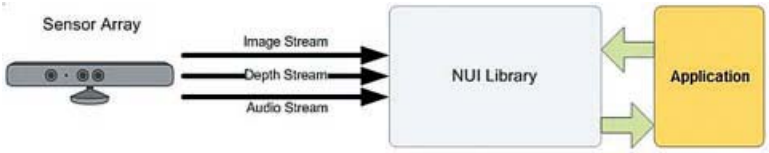


图3　Kinect传感器通过软件库与应用程序进行交互

NUI API概述

NUI，即自然用户界面（Natural User Interface）。NUI API从图像传感器中提取数据，并且控制Kinect设备。NUI API是Kinect API的核心，支持基本的图像和设备管理特性，包括以下几个方面。

- 访问连接到电脑的Kinect设备。
- 通过Kinect图像传感器来获取图像以及深度数据。
- 传送处理过的图像和深度数据，用来支持骨骼跟踪技术。

PC Kinect驱动支持在一台电脑上使用多个Kinect设备。NUI API中包含有计算Kinect传感器数量的函数，你可以决定使用多少个传感器连接到机器上，得到特定传感器的名称，独立地打开和设置每个传感器。

NUI API还提供了更改Kinect传感器组设置的方法，允许从传感器组获得图像数据流。数据流以一系列静态图像的形式传送。一个应用可以从传感器获取以下几类图像数据，

- 色彩数据
- 深度数据
- 玩家分段数据

彩色图像有两种质量水平和两种不同的格式。质量水平决定数据从Kinect到PC的传输速度；色彩格式决定返回给应用的图像数据是RGB还是YUV编码。

传感器组使用USB连接传递数据给PC，并且该连接提供一个给定的带宽值。对图像质量的选择允许调整使用的带宽值。高质量每帧会发送更多的数据但更新比较慢，普通质量的图像更新较快，但由于压缩图像质量会降低。

- 普通质量，彩色图像数据在传递给应用控制台之前会在传感器端进行压缩。接着控制台解

压数据然后再传递到应用上。压缩的使用使返回彩色数据的帧率高达30fps，但是压缩也会导致图像质量的降低。

■ 高质量，彩色图像数据在传感器端不会进行压缩——它将获取的原始数据直接传递给控制台。由于图像没有压缩，那么每帧就要传递更多的数据，最大帧率不会超过15fps。并且没有压缩的数据也需要NUI系统分配更大的缓冲空间。

彩色数据可以有以下两种格式。

■ RGB格式在sRGB色彩空间提供32位线性X8R8G8B8格式的彩色位图。你的应用使用RGB数据必须在打开数据流时指定为NUI_IMAGE_TYPE_COLOR。

■ YUV格式提供16位伽马校正的线性UYVY格式的彩色位图，YUV色彩空间的伽马校正等价于RGB色彩空间的sRGB伽马。由于YUV流每个像素有16位，这种格式保存位图数据时占用较少存储空间，当调用NuiImageStreamOpen时需分配较小缓存。使用YUV数据要求在打开流时指定为NUI_IMAGE_TYPE_COLOR_RAM_YUV。

两种图像格式都从摄像头数据捕获，从而令RGB数据和YUV数据可以表现同一图像。根据应用实现的需要选择合适的图像数据格式。

深度数据流提供了一种结构，在该结构中每个像素的高13位表示在深度传感器的视野内离特定的XY坐标上的物体最近的距离（单位：毫米）。有两种深度数据流可以使用。

■ Frame size of 320X240

■ Frame size of 80X60

应用可以从任意一种深度数据流中处理数据来支持多样的常规特性，例如跟踪用户的运动，识别物体背景可用来在应用播放时忽略掉背景。

在Kinect For Windows SDK Beta中，Kinect系统处理传感器数据来识别出在传感器组前的两个人体图像，然后创建玩家分段图。该图是一张位图，其像素值对应在视野内距离摄像头最近的玩家索引。

尽管玩家分段数据是隔离的逻辑流，实际上深度数据和玩家分段数据合并到了一个单独的结构。

■ 每个像素的高13位表示从深度传感器到最近的物体的距离，单位：毫米。

■ 每个像素的低3位表示在像素的XY坐标系上追踪到可见的玩家的索引。这3位可看做整型值。

玩家索引值为0表示在相应位置没有找到玩家。1和2表示检测到的玩家数量。玩家分段数据常用来隔离特定的玩家或是从原始的彩色深度图像中分离出感兴趣区域。

NUI骨骼API提供了站在Kinect前面至多两个人的位置信息，包括详细的姿势和方位信息。提供给应用代码的数据为组成一个骨骼的点的集合。该骨骼表示了当前玩家的位置和姿势。Kinect最多可以支持20个骨骼点，数据对象类型是骨骼帧形式提供，每个帧里面最多可以保持20个点，见图4。

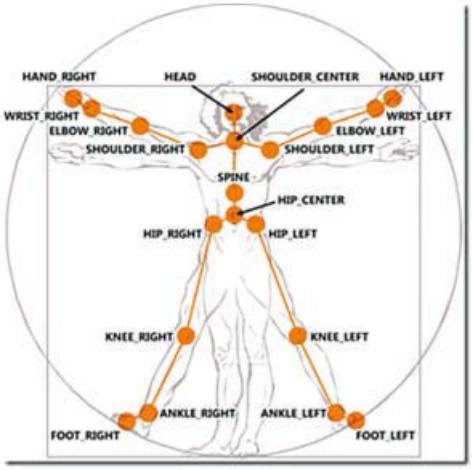


图4 Kinect最多可以支持20个骨骼点

应用代码通过传递一个缓存给NuiSkeleton-GetNextFrame得到骨骼数据。如果最新的骨骼数据准备好了，就会复制到缓存。如果当代码发出请求时，新的骨骼数据还未准备好，那么可以选择等待下一个骨骼数据准备好，也可以立即返回稍后再请求。NUI骨骼API相同的骨骼数据只会提供一次。

NUI骨骼API提供了两个应用模型。

■ 轮询模型

轮询模型是读取骨骼事件最简单的方式。首先应用代码调用NuiSkeletonGetNextFrame，制定等待下一骨骼数据的时间。当新的数据准备好或是超出等待时间，NuiSkeletonGetNextFrame才会返回。

■事件模型

事件模型支持将骨骼数据的获取集成到应用引擎中，这样更加灵活和准确。应用代码传递一个事件句柄给NuiSkeletonTrackingEnable。当新的数据准备好了，便会立即通知该事件。唤醒任一等待的进程，通过调用NuiSkeletonGetNextFrame获取骨骼数据。

Audio API概述

开发应用可以使用Windows Audio Session API从Kinect传感器麦克风组获取原始音频流。开发者可以通过DMO获得DSP。Kinect for Windows SDK Beta包括一个Windows麦克风组DMO的扩展版本，用来支持Kinect麦克风组。

尽管内部细节不同，Kinect音频DMO作为标准麦克风组DMO支持同样的接口，并且以同样的方式工作。然而，KinectAudio DMO：

■有自己的类标识符（CLSID），CLSID_CMSRKinectAudio。

■支持一个附加模式，专门支持Kinect麦克风组。

■包含波形功能，该功能通过一个接口展现，ISoundSourceLocalizer。

Kinect for Windows SDK Beta还包含一个管理音频API，主要是KinectAudio DMO的一个封装，支持同样的功能但是用起来更简单一些。该管理API允许配置DMO和执行操作。管理API还包含提供信号源和波向给应用的事件。

语音识别是自然用户界面的一个关键方面，并且得到Microsoft.Speech平台在Windows下的支持。Kinect for Windows SDK为应用结合Microsoft.Speech API使用Kinect麦克风提供了必要的基础结构，支持最新的语音算法。这个SDK Beta包含了一个定制的声音模型，使得Kinect传感器的麦克风组达到最优效果。

Kinect应用前景

目前已经可以看到国外很多使用Kinect开发出的精彩应用（见图5）。

国内方面，在Kinect for Windows SDK Beta发布伊始，微软亚洲研究院便启动了“微软校园菁英计划”之Kinect Pioneer项目，在全国范围内动员微软学生技术俱乐部的同学集思广益，提交基于Kinect的新创意，并给优秀的创意团队提供Kinect设备和技术支持进行开发。

图5 Kinect精彩应用

Kinect是一个廉价的动作捕捉设备，应用于对动作捕捉精度要求非常严格的领域，这是其未来发展的方向。现在智能手机和平板电脑的发展趋势非常迅猛，同样，Kinect也绝对会有小型化趋势。虽然在专业领域Kinect的技术更新相对消费者领域来说更快，但最后所有技术都汇集到消费者领域。

毋庸置疑，Kinect是一项非常好的技术，但如何为好技术寻找一个好的应用场景，是我们接下来几年之内需要不断探索、不断尝试的方向。

李斌

现就读于西安电子科技大学，微软亚洲研究院学术合作部实习生，Kinect开发者。

吴国斌

微软亚洲研究院学术合作部高校关系经理，负责Kinect在中国的学术研究和项目合作。

体感技术在移动平台上的应用

文 / 吕英阁

2006年，Nintendo发表了新时代游戏主机Wii，掀起了体感游戏的序幕。“体感技术”的基本概念，在于人们可以很直接地使用肢体动作，与周边的装置或环境互动，而无需使用任何复杂的控制设备，便可让人们身历其境地与内容做互动。举个例子，当你站在一台电视前方，假使有某个体感设备可以侦测你手部的动作，此时若是我们将手部分别向上、向下、向左及向右挥，用来控制电视台的快转、倒转、暂停以及终止等功能，便是一种很直接地以体感操控周边装置的例子，或是将此四个动作直接对应于游戏角色的反应，便可让人们得到身临其境的游戏体验。其他关于体感技术的应用还包括：3D 虚拟现实、空间鼠标、游戏手柄、运动监测、健康医疗照护等，在未来都有很大的市场。

体感技术原理及技术演进

如图1所示，图片中的玩家手持游戏手柄进行“网球体感游戏”，玩家的手部击球动作可完全用来模拟并控制游戏里游戏角色的球路，而玩家手部的动作与游戏角色球路的对应是如何实现的呢？原理在于玩家手上的手柄能获取玩家手部的各种物理参数，例如：加速度、角速度、位移……等等，然后再进一步通过算法，将这些物理参数转化为人体在空间中的三个位移量，以及三个旋转量，如此一来便可将手部在空间中的各种动作（平移 + 旋转）完全描述出来，接着再将此平移及旋转量传输给游戏角色，游戏角色便可与玩家做出相同的动作对应。因此所谓的“体感游戏”，便是通过各种传感器捕捉人体的肢体动作（平移 + 旋转），并将所计算出的肢体动作对应于游戏上角色的反应，使玩家的动作与游戏中角色的反应呈现1:1拟真的对应。

图1 3D空间中动作捕捉原理

图2说明了近年来全世界在体感技术上的演进，依照体感方式与原理的不同，主要可分为三大类：惯性感测、光学感测以及惯性及光学联合感测。

惯性感测：主要是以惯性传感器为主，例如用重力传感器，陀螺仪以及磁传感器等来感测使用者肢体动作的物理参数，分别为加速度、角速度以及磁场，再根据此些物理参数来求得使用者在空间中的各种动作。主要代表厂商为Logitech在2007年推出空间鼠标（Mx Air），使用三轴重力传感器以及两轴陀螺仪，可感测使用者在空间中的手部动作，并将此动作转化为鼠标在屏幕上垂直方向与水平方向的位移。2009年，苹果智能型手机开始拉开了手机体感游戏热门下载的序幕，许多使用惯性传感器来适配的体感游戏不断地孕育而生。其中，iPhone使用了以三轴重力感测以及三轴磁传感器为主的惯性感测。2010年，基于未来手机上即将陆续推出拥有重力传感器、磁传感器和陀螺仪的智能型手机，CyWee发展了面向这三种传感器的特有算法，称为九轴混合感测算法（9 - axis Sensor Fusion Technology）。所谓的九轴，指的便是可量测空间中三轴向之重力传感器、可量测三轴向之磁传感器，以及可量测三轴向之陀螺仪，此算法可克服传统上仅使用个别单一传感器的缺点，进而达成更精确的体感体验。

光学感测：主要代表厂商为Sony及Microsoft。早在2005年以前，Sony便推出了光学感应套件——EyeToy，主要是通过光学传感器获取人体影像，再将此人体影像的肢体动作与游戏中的内容互动，主要是以2D平面为主，而内容也多属较为简易类型的互动游戏。直到2010年，Microsoft发表了跨世代的全新体感感应套件——Kinect，号称无需使用任何体感手柄，便可达到体感的效果，而比起EyeToy更为进步的是，Kinect同时使用激光及摄像头（RGB）来获取人体影像信息，可捕捉人体3D全身影像，具有比起EyeToy更为进步的深度信息，而且不受任何灯光环境限制。

惯性及光学联合感测：主要代表厂商为Nintendo及Sony。2006年所推出的Wii，主要是在手柄上放置一个重力传感器，用来侦测手部三轴向的加速度，以及一红外线传感器，用来感应在电视屏幕前方的红外线发射器讯号，主要可用来侦测手部在垂直及水平方向的位移，来操控一空间鼠标。这样的配置往往只能侦测一些较为简单的动作，因此Nintendo在2009年推出了Wii手柄的加强版——Wii Motion Plus，主要为在原有的Wii手柄上再插入一个三轴陀螺仪，如此一来便可更精确地侦测人体手腕旋转等动作，强化了在体感方面的体验。至于在2005年推出EyeToy的Sony，也不甘示弱地在2010年推出游戏手柄Move，主要配置包含一个手柄及一个摄像头，手柄包含重力传感器、陀螺仪以及磁传感器，摄像头用于捕捉人体影像，结合这两种传感器，便可侦测人体手部在空间中的移动及转动。

体感技术于移动装置端最新应用

——九轴体感技术＋无线多媒体串流技术

体感技术应用移动装置上时，往往受限于手机屏幕天生便嵌入手机的特性，因此仅能以较简单的体感方式来操作所有的体感游戏。例如在图3的赛车游戏中，原本使用按键来控制赛车游戏的转向，现在便可透过倾斜手机的方式，来以体感方式真实模拟赛车游戏中的转向。然而，在图4中的高尔夫球游戏，若你同样想要以体感的方式挥动手臂来模拟游戏中人物的挥杆，便会发觉无法实现此操作，原因在于：当你挥动手臂时，手机的画面也随着手臂挥击出，此时便无法观察游戏的进行，因此这类型的游戏十分不适合体感操作。

图3 手机上赛车体感游戏的操作

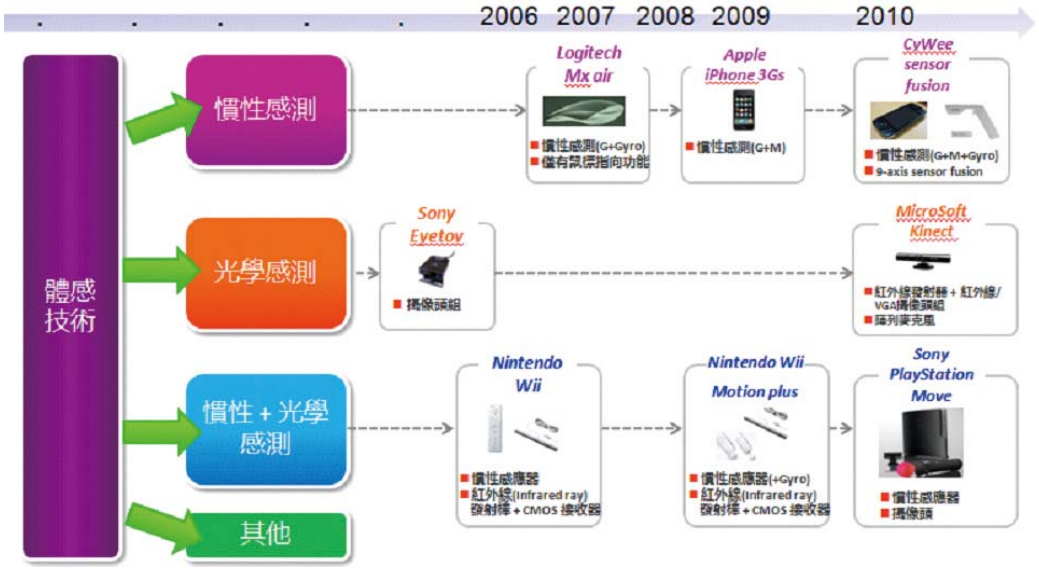


图2 体感技术演进

而如图5所示，若能将手机视频以无线、实时的方式传输至一固定屏幕（TV）上，此时玩家眼睛所及的游戏画面并不会因手机的挥动而影响体感游戏的进行，因此便可在手机上体验

图4 手机上高尔夫体感游戏的操作

图5 在移动设备上将体感与无线多媒体串流结合的全新应用

类似高尔夫球的体感游戏了。这是一个在移动设备上的创新应用，大大扩增体感游戏厂商开发体感游戏上的自由度，而无需局限于仅以较简单体感方式倾斜或转动手机来进行体感游戏的开发。

最适用于移动设备上之体感技术（1）

——九轴惯性感测（9-axis Sensor Fusion）

对于一般移动设备来说，例如手机及Table，皆为可携式组件，因此需有固定于屏幕前方的光学感应棒的体感技术将受限。由于惯性体感感测无需任何感应棒置于玩家前方，因此以惯性感测方式应用于手机体感游戏将是最佳的解决方案。而惯性感测中九轴混合感测算法（9-axis Sensor Fusion），将提供真正1:1动作对应，实时以及 360° 自由动作检测的最佳体现。

使用九轴混合感测算法的原因在于，各个单一的惯性传感器都有各自的缺点，因此势必得使用混合感测的方式，通过彼此补偿来加以改善各自的缺点。如图6所示，如果仅仅只使用重力传感器，将会产生延迟反应、强烈误动作以及奇异点等问题，这是因为重力传感器会同时量测到重力变化以及人体加速度变化，因此若无法将此两个感测值加以分离，便会造成上述问题。而若是以重力传感器加上磁传感器，除了延迟反应之外（磁传感器反应速率远小于重力传感器），磁传感器也易受外界磁场的干扰，而造成感测错误的问题。而若是以重力传感器加上陀螺仪的组合，则会产生累积误差，而且陀螺仪易受温度影响而产生的感测值飘移（drifting）等问题。因此，最完美的组合方案便是将三种传感器结合起来进行混合感测及补偿，便可达到最精确、最实时、绝对位置及不受磁场干扰等的体感体验。

图6 惯性感测中各种传感器特性比较

图7中描述了CyWee九轴混合算法的各种输出，其中的校正后的Sensor原始数据，CyWee提供了动态及静态校正，可让使用者在使用中不会遇到感测数据不精确的问题，同时，也可使得工厂制造端无需额外进行传感器校正程序。以及Orientation、旋转矩阵、四元素、 重力变化以及线性加速度输出，也都提供了体感游戏开发商直接调用。例如OpenGL定义的3D物体旋转及平移的API，节省了游戏开发商对于体感演算开发的时间，这些输出已经被定义于Google Android 2.3之后版本的API中。此外，

CyWee九轴混合算法所提供之输出	说明
校正后之sensor原始资料	动态及静态校正
Orientation	Roll, Yaw, Pitch角度－提供实时,绝对位置
旋转矩阵	空间中3D物体旋转量
四元素输出	空间中3D物体旋转量
重力变化	物体感受空间三轴向重力变化
线性加速度	物体感受空间三轴向直线加速度变化
动作判断输出	360度, 3D空间全向动作识别

图7 CyWee Sensor Fusion算法输出内容

CyWee也提供了各种动作判断输出，可供体感游戏开发商直接调用。

图8描述了CyWee为开发商提供体感游戏中各种体感动作算法的输出结果，称为动作数据库（Motion Library），可大大地节省体感游戏开发商开发体感游戏的时间及成本，根据不同的动作类型，可适用于不同类型的体感输出，例如射击类动作输出、射箭类动作输出、击打类动作输出等。

图8 CyWee 动作数据库（Motion Library）可直接适于用各种体类型之体感游戏

最适用于移动设备上之体感技术（2）

——无线多媒体串流技术（Wireless HDMI）

图9描述了世界上各个厂商对于无线多媒体串流技术的规格比较，可根据操作频段、影像压缩方式、功耗、传输带宽、支持最大分辨率、延时时间以及传输距离等因素来进行比较，可发现应用于移动设备上时，功耗及传输所需带宽为最大的问题。

功耗方面由于移动设备为一使用电池的装置，因此，在能够让使用者使用无线多媒体串流超过数个小时（例如：玩游戏、看影音视频）而不会耗尽电池的各种方案中，可发现CyWee的解决将是最佳的解决方案。而在所需传输带宽方面，由于手机使用的是1（Transceiver）X 1（Receiver）天线，因此所能支持之最大传输带宽有限，所以在超过 1 X 1 天线所能提供带宽的解决方案，都将是无法适用的。

最新动态与未来展望

CyWee已拥有多年的发展体感技术的经验，除了原本在体感游戏及体感手柄销售上有长足的进展外，在移动领域上更是为之前JIL

	WiDi Intel Wireless Display	WHDI Amimon	WirelessHD SiBEAM	CyWee
Band	5G	5G	60G	2.4G+5G
Data Compressed	Yes	No	Yes	Yes
Power Consumption	>3W	>6W	>11W	<0.15W chip .5W system
Bandwidth	100M (1x1)	1.5G (4x5)	5G	40M (1x1)
Resolution	720P, 30fps	1080p , 30fps	1080P , 60fps	1080P, 30fps
Latency	600ms	1ms	10ms	<1ms
Distance	30m	30m	10m	30m
Multicast	Yes	No	No	Yes

图9 无线多媒体串流技术比较

（Joint Innovation Laboratory）Project所唯一指定的体感方案供货商。目前更进一步的授权许多国际大厂使用CyWee体感算法，并于Android 2.3版本上加以整合，以适配Android 2.3 版本之后关于体感方面API的定义。

而在无线多媒体串流的领域，由中国移动所主导的WiMo（Wireless Mobile Multimedia Transmission Protocol）移动终端无线多媒体传送技术，为支持高清音视频无线传输的技术协议，该技术支持移动终端与大屏幕设备之间无线、实时、高画质视频信息传输，可实现以行动终端为核心的多屏共享互动。

展望未来，九轴体感技术结合无线多媒体串流技术在移动设备上的发展潜力是值得期待的，而无线多媒体串流技术除了会更进一步提升芯片的性能外，也会将其整合于更多需要高传输率、低延时视频压缩的平台中（例如: 网络视频播放、云端游戏等），相信最终必定能够在各种移动的平台上，创造出许多属于未来的应用。

吕英阁

2003年获台湾大学机械系硕士学位。大学毕业后，进入台湾工业技术研究院微系统中心，从事体感技术研究。2009 年加入速位互动股份有限公司（CyWee Group Limited）， 继续从事体感技术的研究，作为核心研发成员，完成了9轴体感技术的核心算法。

TCL Android开发者大赛成果即将发布

TCL Android开发者大赛面向所有热爱研究、设计、开发安卓应用的团体和个人，从4月2日开始至8月15日结束，已收到超过1700件作品。

与以往活动重金悬赏不同的是，此次开发者大赛更注重对应用开发者的扶持以及如何帮助应用成长。除提供实物奖励外，更侧重向优秀安卓开发者提供推广的渠道和平台，增加产品曝光度和下载量，为参与的开发者提供广阔的创业平台。

业内专家认为，智能电视市场大热，目前国内各大彩电厂商也将相继开放自有平台，以吸引更多第三方开发者。日前，由TCL、长虹、海信三家智能终端龙头企业组建的中国智能多媒体终端技术联盟（简称“中智盟”），对外正式发布中智盟应用程序商店技术标准SDK，为应用开发商搭建强大的开发环境，为终端消费者提供简单的界面，为业内的各厂家开辟一个智能终端合作的空间。这在未来将是继续吸引更多的开发者参与应用程序商店的作品开发积聚用户平台。中智盟近期的三大目标是：应用程序商店技术标准、三屏幕互动技术标准、TVO/S。中智盟争取2~3年内培育一万家以上独立的软件开发商，打造十万个以上的精品应用。

随着技术的进步和网络的融合发展，以及以体验为核心的商业模式不断创新，各网络间互联互通，业务运用上互相渗透，实现网络资源最大限度的共享，将成为满足社会发展需求和消费者利益的根本出发点。未来的TCL将通过技术和用户体验的创新，不仅使智能终端开始有了统一的技术标准，满足更多应用开发者和消费者的需求，更符合智能电视产业开放共赢的未来产业新格局，同时也宣告国内骨干企业通过合作创新模式领跑智能电视产业，开启了让消费者体验优质生活的新篇章。

业内分析师认为，我国彩电业的转型升级有较高的压力。在技术更新换代、市场激烈竞争的环境下，提高我国彩电业的发展速度和质量，是与时俱进的举措，必将满足消费者的多样化需求。智能电视的发展正是增强我们彩电业内涵的最主要途径，也是历史赋予彩电业的责任。智能电视，符合产业融合发展的大趋势，是彩电业未来发展的重要方向，具有广阔的市场前景。对于满足市场的需求，加强信息技术的应用，增强技术创新的能力，加快转变升级，具有较高的意义。

主持人：张银奎

《软件调试》一书作者，从事软件开发和研究10余年，对IA-32架构、操作系统内核、虚拟技术，尤其对软件调试有较深入的研究。翻译（合译）作品包括《数据挖掘原理》、《机器学习》、《人工智能：复杂问题求解的结构和策略》、《观止——微软创建NT和未来的夺命狂奔》等。

漫谈Android系统的调试模型（上）

本文以Android应用程序调试模型为基础，四个进程为线索，深入分析了Android系统的调试方法。

说到Android，我第一次接触它的时间并不算晚，回想起来，刚好是三年前的夏天，今天Android的流行势如破竹、不可阻挡。为了追赶一下流行，这一期我们也来谈一下Android！谈什么呢？当然还是谈调试。

架构一览

让我们直接看一下Android的应用程序调试模型，图1画出了有关的各个部件以及简化了的相互关系，左侧是运行Android的目标系统，右侧是用于开发和调试的主机，这里以Windows为例。

从这个调试模型来看，调试Android应用程序需要两台机器，被调试程序运行在目标机上，调试器运行在主机上。这样做主要有两个考虑，首先是Android系统可能运行在手机等显示屏幕较少甚至没有图形显示的系统上，而调试器程序通常需要较大的显示空间。其次是Android的图形系统比较简单，不可以同时显示多个程序的窗口，如果显示了被调试程序的窗口，那么就无法同时显示调试器的窗口。这使得调试程序和被调试程序运行在同一台Android系统中有很多实际困难。此外，双机调试也是嵌入式系统中惯用的方式。

图1中并排的四个矩形代表四个进程，依次是运行在Java虚拟机中的被调试Java程序进程，运行在后台的ADB监护程序（ADB Daemon），运行在主机上的ADB服务程序，运行在主机上的Eclipse开发和调试环境进程。ADB是Android Debug Bridge的缩写。ADB监护程序与ADB服务程序通过USB电缆或者网络传递数据，它们协同配合架起一座桥梁，让运行在主机上的调试器可以自由访问和控制运行在目标机中的被调试Java进程，如同调试本地的程序一样。二者的距离可以近在咫尺，也可以远隔千里。下面将以这四个进程为线索分别做一些深层描述。

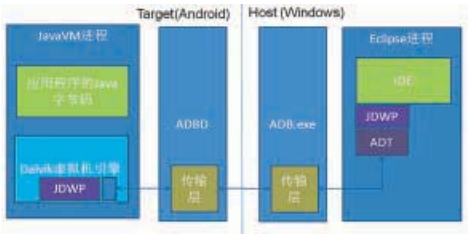


图1 Android系统的应用程序调试模型

特立独行的Dalvik

特立独行，何其难也。但唯有特立独行，才能不同凡响。Dalvik便是一个典型的例子。

Android系统中的大多数应用程序都是用Java编写的，包括很多内建的应用程序，按Alt+F1打开一个控制台窗口（Alt+F7返回图形桌面），然后使用ps命令观察Android系统的进程，可以看到很多进程都是Java进程。以图2所示的执行结果为例，进程名（最右一列）以“com.”开始的进程都是Java进程。

但值得强调的是，这么多Java进程中

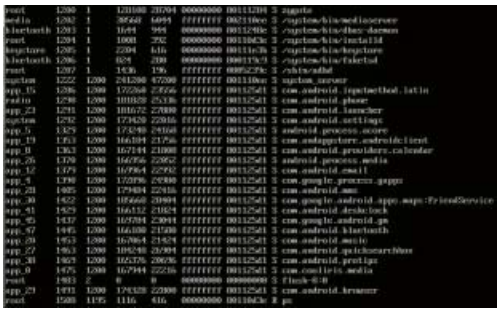


图2 观察Android系统中的进程(局部)

并没有运行普通的Java虚拟机，而是运行着一个特殊的Java虚拟机，这个虚拟机的名字便叫Dalvik。

Dalvik最初是由Dan Bornstein一个人编写的，这个名字源于Dan的祖先曾居住过的一个冰岛小渔村。Dalvik是专门为计算能力较差、内存较少的系统而设计的。与普通的Java虚拟机相比，它有很多特立独行之处。今天的CPU大多都是基于栈的，是所谓的Stack Machine，普通的Java虚拟机也是基于栈的。而Dalvik不是，它使用的是所谓的Register Machine，基于寄存器。另外，Dalvik的字节码定义和执行文件的格式（.dex）也是不一样的。甚至在2.2（代号Froyo）之前的版本中连及时编译技术（JIT）也没有采用。

因为专门针对资源有限的系统设计，而且针对特定的硬件平台做过精心的优化，当用户使用Android手机时通常不会感觉运行缓慢。在用户体验至上的今天，这一点对于Android系统的成功是非常重要的。

标准的调试协议

Dalvik有很多独到之处，但是在调试方面，为了支持标准的调试器，它沿用了标准Java的做法。图3是Java平台调试器架构（JPDA）的示意图，选自《软件调试》一书的补编内容。比较图1和图3，可以看到二者很是神似。

从图中可以看出，JDPA由以下三部分组成：

- Java调试器接口（Java Debug Interface），简称JDI，这是一套供调试器或者性能分析工具使用的Java API，用来访问目标程序的内部状态和调用Java虚拟机的各种调试功能。JDI工作在调试器进程中，负责与调试器的其他部分进行交互。
- Java虚拟机工具接口（JVM Tool Interface），简称JVM TI，它是JVM对外提供调试服务的标准接口，工作在被调试的Java进程中，负责与Java虚拟机进行交互收集调试信息并接收和处理来自JDI的命令请求。与JDI是100%使用Java语言开发的Java库不同，JVM TI是一个本地的（native）编程接口，通过这个接口，工具软件可以观察Java虚拟机（JVM）中所执行程序的状态并控制它的执行，以实现调试、Profiling、监控、线程分析、覆盖率分析（Coverage Analysis）等目标。
- Java Debug Wire Protocol，简称JDWP，这是JVM TI与JDI之间进行通信的协议，二者通过这个协议交换信息。

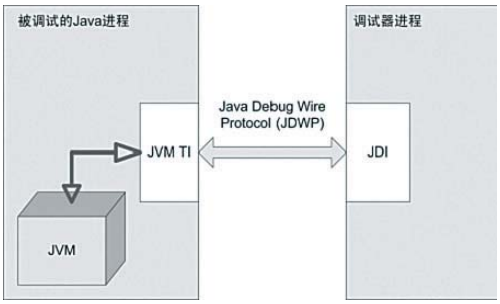


图3 Java平台调试器架构（JPDA）

通过如下链接，可以打开JDWP的详细规约网页：http://download.oracle.com/javase/1,5.0/docs/guide/jpda/jdwp/jdwp-protocol.html。

下面来看一下Dalvik中是如何实现以上标准的。首先来看JVM TI，从源文件角度来看，这个部件主要是实现在如下源文件中的：dalvik\vm\Debugger.c。

在这个文件开头的注释明确描述了这个文件的作用“Link between JDWP and the VM”。文件中实现了一系列以dbgDbg开头的函数。虽然这些函数没有完全按照JVM TI所定义的函数原型来设计，但是从功能上来看，二者是一致的。例如，JVM TI定义了如下两个函数来获取所有线程和挂起指定的线程：

```
jvmtiError
GetAllThreads(jvmtiEnv* env, jint*
threads_count_ptr, jthread** threads_ptr)
```

```
        jvmtiError
        SuspendThread(jvmtiEnv* env,
jthread thread)
```

相应的，在Debugger.c中，可以看到如下两个函数与其对应：

```
void dvmDbgGetAllThreads(ObjectId**
ppThreadIds, u4* pThreadCount)
void dvmDbgSuspendThread(ObjectId
threadId)
```

JDWP的具体实现

下面详细分析一下Android系统（Dalvik）对JDWP的实现，有关的源文件有很多个，大都位于如下文件夹中：dalvik\vm\jdwp\。

其中较重要的文件有：

- JdwpMain.c：这一子模块的主文件，负责初始化（dvmJdwpStartup函数）、停止（dvmJdwpShutdown）、获取状态（dvmJdwpIsActive）等功能。在初始化函数dvmJdwpStartup中，会准备传输层（下一节详细描述），并启动一个工作线程。
- JdwpAdb.c：通过ADB监护进程进行通信的传输层。
- JdwpSocket.c：通过TCP网络进行通信的传输层。
- JdwpHandler.c：处理来自调试器的请求。
- JdwpEvent.c：主动向调试器发送调试事件，典型的调试事件有线程启动和停止、类加载和卸载、异常等。

Android实现了两种传输方式来支持JDWP对话，一种是TCP网络，另一种是通过USB电缆。当使用USB电缆时，Android设备端应该有USB OTG类型的USB端口，也就是设备端的端口，以便与主机上的主机端口连接，因为USB总线是树形拓扑结构，主机端口和设备端口可以相互通信，而主机端口和主机端口是无法直接通信的。

在JDWP的主文件JdwpMain.c中，初始化函数dvmJdwpStartup会根据参数结构（JdwpStartupParams）中的transport字段（pParams->transport）来决定使用何种传输层。

```
switch (pParams->transport) {
    case kJdwpTransportSocket:
            // LOGD("prepping for JDWP
over TCP\n");
    state->transport =
dvmJdwpSocketTransport();
        break;
case kJdwpTransportAndroidAdb:
          // LOGD("prepping for JDWP
over ADB\n");
state->transport =
dvmJdwpAndroidAdbTransport();
        /* TODO */
        break;
```

而这个参数是由虚拟机启动函数层层传递过来的，虚拟机启动函数又是从命令行中的-Xrunjdwp或者-agentlib:jdwp参数中获取的，比如下面的参数值便是选择TCP方式的传输层：

```
"-Xrunjdwp:transport=dt_socket,add
ress=8000,server=y,suspend=n"
也可以写为：
"-agentlib:jdwp=transport=dt_socke
t,address=8000,server=y,suspend=y"
```

其中的address=8000,server=y的意思是让被调试端充当服务器角色，等待调试器来连接，首先尝试监听8000端口，如果失败，那么便继续尝试8001，8002等等。另一种的工作方式是让被调试端直接连接已经在等待的调试器，也就是充当客户，这时要把配置参数写为

```
server=n,address=<hostname:port>,
```

其中的hostname:port用来指定调试器所在的主机和所监听的端口。

下面是来自Dalvik调试文档（dalvik\docs\debugger.html）的两个例子。

例子1：让被调试进程使用ADB传输层。

```
% dalvikvm
-agentlib:jdwp=transport=dt_android_
adb,suspend=y,server=y -cp /data/foo.jar
Foo
```

例子2：让被调试进程使用TCP网络传输层，然后让ADB进程转发TCP连接。

```
% adb forward tcp:8000 tcp:8000
% adb shell dalvikvm
-agentlib:jdwp=transport=dt_socket,addr
ess=8000,suspend=y,server=y -cp /data/
foo.jar Foo
```

启动调试器：

```
jdb -attach localhost:8000.
```

在建立好传输层后，dvmJdwpStartup会调用dvmCreateInternalThread（&state->debugThreadHandle, "JDWP", jdwpThreadStart, state)创建一个新的线程专门负责JDWP会话。因为这个原因当我们使用ps –t命令显示线程列表时，在每个Java进程中，都可以看到一个名为JDWP的线程，如图4所示。

图4中显示了两个Java进程，一个是

本期问题：

上期答案是：文件描述符和文件句柄是否是一回事，答案为不是，前者是Unix和Linux系统中的术语，也广泛用在POSIX函数中，后者来自Windows系统，二者虽然都是通过一个整数来关联一个内核对象，但是最好区别对待。以open函数为例，它的返回值是文件描述符类型的，Windows也支持这个函数，其实现是调用CreateFile打开文件得到操作系统层的文件句柄，然后再调用_alloc_osfhn在全局数组__pioinfo中寻找一个空闲元素或者分配一个新的ioinfo结构，然后把操作系统句柄保存到这个结构中，最后把数组元素的序号作为文件描述符返回。

本期的问题是：你知道Android的属性系统是如何实现的吗？

编者说明：

- 投稿信箱：contest@csdn.net
- 联系方式写在一个单独的TXT文件里，包括以下几项：
  1）姓名
  2）工作单位或学校
  3）电话联系方式
  4）邮寄地址
  5）E-mail地址
- 解答提交时间，最好早于当月15日。

deskclock，另一个是media，它们的第三个线程都是JDWP线程。

图4 Java进程中的JDWP线程

设备端ADB监护进程（ADBD）

下面我们再把目光转向前面提到过的ADB进程，因为目标端和主机端都有ADB进程，所以我们分别来叙述，先来看设备端的ADB监护简称，因为它的可执行文件名叫adbd（小写），所以我们就用ADBD来称呼它，以便行文简洁。

首先，ADBD进程是由系统的init进程根据init.rc文件的指示而创建的，以下便是来自init.rc的有关内容：

```
service adbd /sbin/adbd recovery
disabled
on property:persist.service.adb.enable=1
        start adbd
on property:persist.service.adb.enable=0
        stop adbd
```

第3行和第5行的意思是在Android的属性系统中查询persist.service.adb.enable属性，这个属性的默认值保存在系统根目录的default.prop中，值为1，因此，默认情况下，ADBD总是启动的，而且一旦被杀掉，便会再运行一个实例，因此在典型的Android系统中，我们总可以看到ADBD进程，比如在图2所示的进程列表中，上数第7行便是ADBD进程，它的可执行文件位于/sbin目录中。

ADBD程序的源代码位于如下目录中：system\core\adb\。

从这个路径也可以看出，ADB在Android系统中有着很重要的地位，是以核心部件的身份存在的。

ADBD程序的入口函数main位于adb.c文件的末尾，只有几行代码，其中一个动作是调用start_device_log(void)函数初始化日志文件。日志文件是可配置的，如果系统属性persist.adb.trace_mask的值为1，那么便创建/data/adb子目录，并在其中创建log文件。文件名是动态的，形式为：/data/adb/adb-%Y-%m-%d-%H-%M-%S.txt，然后把标准输出(stdout)和标准错误(stderr)重定向到这个文件。如果属性值不存在或者不等于1，那么便直接返回了，不产生log文件。

默认情况下，不会产生log文件，但可以通过如下命令设置trace_mask属性，然后终止ADBD，让系统重新创建ADBD进程，便可以看到日志文件了。

```
# setprop persist.adb.trace_mask 1
# kill <adbd进程号>
# cd /data/adb
# ll
```

处理好日志逻辑后，main便把执行权交给了adb_main函数做更多的初始化动作，其中最重要是：

- 准备监听所谓的smart socket端口（ADB_PORT），端口号为5037，以等待来自客户端的服务请求，与源代码相同目录的services.txt列出了目前已经支持的所有服务，大约有几十种，比如shell、jdwp、sync等。我们在主机端通过执行adb命令所做的很多动作都是通过请求相应的服务而实现的。
- 准备传输层，有两种方式，一种是通过USB线缆，另一种是通过TCP网络，对于后者，会先创建一个新的线程，然后开始监听5555端口（ADB_LOCAL_TRANSPORT_PORT），如果5555端口已经被占用，那么便尝试5557，5559，依次类推。

图5显示了在Android系统中执行netstat命令观察到的端口监听状态。

图5 网络端口状态（目标端）

至此，设备端准备就绪，开始等待来自主机端的连接请求，以便开始对话，我们将在下一期继续介绍。

程序员的利器——Spread表格控件

作为一名程序员，你对第三方控件的期望是什么？许多人的第一反应是易于使用。你一定不愿意在使用前先花几天时间啃产品手册，而总是希望控件立即就能用到项目中。从1991年发布Spread COM 1.0开始，GrapeCity （及FarPoint）在过去二十年一直在为全球的程序员提供能“立即使用”的电子表格控件。虽然随着版本和平台的升级，Spread的功能在不断增强，但程序员仍可以不阅读手册就能使用它的大部分功能。

从外观上看，表格控件只是在窗体（Form）或网页上增加一个看起来像Excel的表格，但实际上不仅仅如此。程序员对可编程的表格控件的需求与表格应用程序大不相同，主要区别是表格控件需要分别提供设计时、编程接口和运行时支持。而Spread在这三种模式下都易于使用。当你创建一个新工程时，可以很简单地拖拽一个Spread的实例到窗体上，右键点击控件，在弹出的菜单中选择Spread设计器。使用Spread设计器就像使用Excel一样方便，可以定制控件的绝大部分属性，如表格的大小、外观、行列数、单元格类型等。

可以把Spread作为一个电子表格嵌入到应用程序中，实现类似Excel的功能（见图1）。用户可以根据任何一列进行排序，自动合并相同内容的单元格，支持渐变填充、背景图像、边界设置、条件格式等样式组合。Spread的公式计算引擎支持300多种内置函数，包括日期、时间函数、工程计算函数、财务计算函数、逻辑函数、数学和三角函数、统计函数、文本函数等，可以导入和保存Excel格式的数据文件，在Spread和Excel之间进行数据的拷贝和粘贴操作。Spread还支持20多种预定义的单元格类型，包括按钮、图片、复选框、组合框、条码和多种常用的数据类型。

Spread强大的定制能力基于“以单元格为中心”的设计理念，能够定制任一单元格的几乎所有属性，包括单元格类型本身。不仅可以定制Spread的外观，还能定义用户如何与Spread进行交互，甚至让用户根本感觉不到他在操作一个功能强大的表格控件。如果你希望以幻灯片的方式展示公司的团队活动照片，并在照片旁显示时间、照片说明等信息，只需把Spread定制为一行多列，然后使用图片单元格。如果希望添加相册效果，可以使用Spread的多个表单等。

现在开发任何管理系统，如销售管理系统、库存管理系统、产品管理系统等，如果你只打算显示数据本身，说明你已经Out了。现在必须用图表化的方式来展示数据，让隐藏在数据背后的规律一目了然地显示在屏幕上。Spread支持85种丰富多彩的图表效果，可在Spread设计器中基于表单数据直接生成图表，操作简单。同时，程序员也可通过代码创建完成图表的数据绑定和类型设置，并对图表的细节进行定制（见图2）。

借助Spread设计器，你可以完成许多功能的设计开发而无需编码。但作为一名程序员，你迟早会发现需要与Spread的编程接口打交道。Spread API采用面向对象的层次结构

图1

模型，查找和使用都很直观。例如，下面的代码用于获取Spread第一个表单的A1单元格：

```
fpSpread1.Sheets[0].Cells[0,0]
```

从这里可以获取或设置这个单元格的所有属性。除了单元格对象外，也可以用类似的直观方法获取行和列的对象。

过去国内的程序员主要使用Spread英文版，难免会有一些不便。2011年8月，葡萄城发布了Spread for Windows Forms 5.0中文版，有效解决了程序员使用国外优秀控件所面临的语言问题。在Spread中文版中，开发人员的交互界面和提示信息都是中文，包括菜单和对话框等交互界面，在设计时控件各对象的方法和属性的说明，以及在Visual Studio环境中的API智能提示等。Spread中文版还包含了近100万字的中文帮助文档，不仅对产品的每一功能进行了细致的介绍并提供示例代码，而且对所有的API都提供了中文的说明和参数描述，便于程序员的学习和使用。

欢迎访问葡萄城控件产品网站http://www.gcpowertools.com.cn 下载试用Spread中文版。

图2

GUI单元测试

文 / Gonen Israeli

本文描述了在TDD C#项目中使用的解决GUI单元测试问题的方法。

在过去几年内，单元测试在软件开发过程中发挥了重要作用。它曾被认为是一项负担和杂活，是在执行后（且有时间时）才做的工作。然而现在人们广泛认为，单元测试是开发过程的一个重要组成部分。单元测试是对类或功能等模块的测试。它具有一些属性，使其区别于其他类型的软件测试。它通常由开发工程师执行，主要关注在测单元（UUT，Unit Under Test）的正确性，而不是与其他单元的测试完整性。

单元测试和测试驱动开发

近期，极限编程等敏捷软件开发方法以及测试驱动开发（TDD）等做法的流行使得单元测试成为人们关注的焦点。测试驱动设计是一种在编写代码前制定代码测试方案的设计技术。它强调创建多个单元测试，以达到完全代码覆盖。TDD还强调单元测试应只测试在测单元，而不是它交互影响的其他模块。在TDD中，单元测试应运行较快（数秒钟），无需执行测试的开发工程师进行手动设置和清理。

这意味着如果我们正在开发的单元需要调用另一个单元，比如数据库写入器类，为了进行单元测试，我们将用伪程序或模拟程序来替换被调用的单元。这将确保测试运行较快，且执行测试的开发工程师无需浪费时间等待数据库许可证或磁盘访问时间。

单元测试和测试框架

单元测试框架使得单元测试的运行和编译更加容易。单元测试框架的例子包括JUnit、CPPUnit、csUnit和Visual Studio单元测试。这些基本上都是测试运行器，它们运行开发工程师编写的一系列测试功能，并按照通过和失败测试的数量汇总结果。在这些测试中，开发工程师通常创建一个单元实例，并通过各种输入进行测试，看它是否能返回预期结果。单元测试框架具有下列特点：测试设置和清理、标准化陈述或检查错误信息。

正如软件开发中经常出现的那样，这一问题有多种可能解决方案，但没有一种是完美的。本文 将描述一种我们在Nova测量仪器公司的TDD C#项目中使用的解决GUI单元测试问题的方法。我将详细描述该方法，描述其优点和缺点，并提出一些备选方案。

首先，需要准确确定我们想单元测试什么。作为单元测试开发工程师，我们应真正将精力放在测试当某个UI小部件（按钮、文本框等）被点击或加载时会发生什么的逻辑上。小部件的颜色、位置和字体会经常变化。检查它们是否正确最好留给质量保证测试（QA）来做。

GUI逻辑测试为什么困难？

第一个问题是GUI模块内存在GUI构建代码和逻辑的组合。分层是软件开发中一个非常重要的概念。我们通常会明智地将我们的代码分成数据、逻辑和UI层。但构建GUI菜单和屏幕的代码经常包含某些逻辑，如当按下一个按钮时做什么、字段内哪些值是合法的等。

第二个问题是按钮和字段等GUI小部件对于用户可见且可访问。因此，对于用户，它们主要是公共接口。这些小部件（按钮等）的变量在包含GUI的类内通常是私有字段。这意味着代码和用户之间小部件接口的可见性通常是不匹配的。

解决方案：

那么，我们想做的是将GUI模块分成两个单独的模块。第一个模块基本上是纯UI，它只能构建菜单和屏幕。第二个模块拥有所有逻辑，如当点击按钮时做什么。我们将把逻辑从GUI模块中分离出来，只留下一个仅能构建UI小部件的超薄层，将所有关于小部件显示什么或当点击或修改小部件时做什么的决定留给单独的UI逻辑模块。

问题示例：

作为示例，我们看一下下面的Java Swing类。它创建一个带按钮的简单Swing面板，对点击按钮的次数进行计数并显示在标签上

下面是面板类的部分java代码。

```
public class CounterPanelNonTdd implements
ActionListener{

    private int numClicks = 0;
     final JLabel label = new JLabel("Num
Clicks: 0");

    public void actionPerformed
(ActionEvent e) {
        numClicks++;
          label.setText("Num Clicks: " +
numClicks);
    }

    public Component createComponents() {
         JButton button = new JButton("I'm
a Swing button!");
        button.addActionListener(this);
          //Next add the button and the
label to a new panel
          //and return the panel to the
caller . . .
```

下面是创建面板并在窗口中显示的部分代码。

```
CounterPanelNonTdd app = new
CounterPanelNonTdd();
Component contents = app.
createComponents();
// contents is the GUI panel
// now we can display it in a window
frame
```

解决方案：将面板类分成UI和逻辑：

我们关于面板类的主要问题是它将UI创建和UI逻辑组合在一起。计数器和增大计数器的决定不属于创建小部件的UI类。我们需要将其移至单独的UI逻辑类中，从而更容易利用传统的单元测试方法进行测试。

薄UI面板类：

下面是我们第二版面板UI类。我们已经将所有逻辑从面板移至一个单独的逻辑类中。UI逻辑类知道要在标签中显示什么。UI面板类拥有标签。现在，UI逻辑类需要告诉UI面板类要显示什么。为此，我们将所有小部件置于UI面板类将执行的一个公共接口上。

```
public class CounterPanel implements
ICounterPanel {

    JLabel label;

    public void setLabelText(String
labelText) {
label.setText(labelText);
    }

    public Component
createComponents(ActionListener
counterButtonListener)
    {
         label = new JLabel();
        JButton button = new
JButton("I'm a Swing button!");
        button.addActionListener(counter
ButtonListener);
        //Next add the button and the
label to a new panel
        //and return the panel to the
caller . . .
```

请注意，该代码不再拥有计数器或增大计数器的决定。我们已将所有逻辑删除，并将所有决定交给外部。该类构建一个按钮，但不决定当点击按钮时做什么。它只是将该决定转交给UI逻辑类。它还创建标签，但不决定标签显示什么，而是将这一决定再次留给UI逻辑类。现在，这个UI面板类基本上是一个空壳，它只是创建小部件，并将其与UI逻辑类连接。它现在已没有价值，我们将不会对它进行单元测试。我们将把菜单是否处于适当位置的目视检查留给QA。

（请注意，如果面板构建代码由可视GUI设计器生成，则我们对UI面板类代码的修改可能在小部件移至设计器上且类代码重新生成时被覆盖。C#的Visual Studio设计器通常会保留我们的手动修改，但并非总是如此。）

创建UI面板类并在窗口中显示的代码现在如下：

```
ICounterPanel counterPanel = new
```

```
CounterPanel();
CounterPanelUILogic counterPanelUILogic =
new CounterPanelUILogic(counterPanel);
Component contents = counterPanelUILogic.
Initialize();
// contents is the GUI panel
// now we can display it in a window
frame
```

在前面不可测试的示例中，主程序从面板类的createComponents函数得到实际面板。现在，主程序从UI逻辑类的初始化函数中得到实际面板。主程序创建UI面板类和UI逻辑类。然后，它通过将UI面板类作为一个参数传递给UI逻辑类使它们发生联系。

UI逻辑类：

```
public class CounterPanelUILogic
implements ActionListener{

    private ICounterPanel counterPanel;
    private int numClicks = 0;

    public Component Initialize()
    {
       Component panel = counterPanel.
createComponents(this);
       counterPanel.setLabelText("Num
Clicks: " + 0);
       return panel;
    }

     public CounterPanelUILogic(ICounterPa
nel counterPanel) {
       this.counterPanel = counterPanel;
    }

    @Override
           public void
actionPerformed(ActionEvent e) {
        pressCounterButton();
    }

    public void pressCounterButton() {
       numClicks++;
        counterPanel.setLabelText("Num
Clicks: " + numClicks);
    }
}
```

该类包含UI的所有逻辑代码，它是我们将进行单元测试的类。我们先看一下要点。初始化函数告诉UI面板类创建面板、按钮等，将UI逻辑类本身作为按钮监听者。当按下按钮时，actionPerformed函数将被调用，进而调用pressCounterButton函数，点击次数将增加。最后，UI逻辑类将告诉UI面板类将标签设为新的点击次数。

为什么UI逻辑构造器得到一个ICounterPanel接口参数，而不是counterPanel类参数？为什么actionPerformed不会使按钮增大，而是要调用pressCounter公共函数？

答案是单元测试。在产品代码中，我们将向UI逻辑类发送一个真实的面板类，但在单元测试中，我们将发送一个代替它的模拟程序，使得单元测试更加方便和快捷。

有点不太现实，但更快、更易测试和更好的设计。

该类只模拟实际面板类。优点是更快，不会使窗口飞过屏幕，我们的测试在按钮改变位置、名称或文本时不太可能中断（需要修改）。缺点当然是我们只模拟实际图形以测试逻辑，因此，我们并未真正检查实际小部件是否真正收到具体的值。我们将此项工作留给QA。

这种解决方案的另一个优点是它实际上改善了我们的设计和代码。我们现在将UI和逻辑更好地分离，这意味着我们的代码更好地进行了分层、更加模块化和更加稳定。例如，我们可以利用同一UI逻辑类在多个具有不同外观和感觉的不同UI面板类中进行发送。我们还可以通过键盘快捷方式、记录和自动播放功能等方法调用pressCounterButton函数。在单元测试中，如果要运行得更快，我们总是愿意让测试变得不太现实，关注我们的单元，并改进我们的设计。

替代解决方案：

一种替代解决方案是把逻辑和UI组合在同一类中。然后，我们在测试中所能做的就是将所有小部件暴露为该类的一个公共接口。虽然有些人认为这对封装造成不必要的破坏，但我认为，在用户看来，小部件已经成为公共接口的一部分，因此，使得小部件访问对于模块外部变成公共的，这也是正确的。

```
public class CounterPanelTesting2
implements ActionListener{

    private int numClicks = 0;
     final JLabel label = new JLabel("Num
Clicks: 0");
    JButton button;

    public String getLabelText()
    {
       return label.getText();
    }

    public void clickCounterButton()
    {
       button.doClick();
    }
public void actionPerformed
```

```
(ActionEvent e) {
        numClicks++;
          label.setText("Num Clicks: " +
numClicks);
    }

    public Component createComponents() {
         button = new JButton("I'm a Swing
button!");
        button.addActionListener(this);
          //Next add the button and the
label to a new panel
          //and return the panel to the
caller . . .
```

我们可注意到此版本测试代码的一些不同。

```
@Test
 public void TestCounterPanel2()
 {
  CounterPanelTesting2 counterPanel =
new CounterPanelTesting2();
  Component contents = counterPanel.
createComponents();

  Assert.assertEquals("Number of button
clicks: 0", counterPanel.getLabelText().
trim());

  counterPanel.clickCounterButton();

  Assert.assertEquals("Number of button
clicks: 1", counterPanel.getLabelText().
trim());
 }
```

第二个解决方案的不同之处：

该解决方案看起来是测试更真实的代码。它实际上在真实标签小部件中获得和设置结果。这意味着我们现在需要调用修整函数，以消除标签字段多余的白色空间。

它强制将更多小部件暴露为该类公共接口的一部分。如果小部件改变类型，特别是当我们返回原始小部件时，该解决方案更有可能失败。例如，我们可能选择返回标签本身，而不是其文本，这将使代码更加脆弱且易受变化的影响。该解决方案还依赖于图形库的支持。它依赖Java doClick函数来强制从单元测试发送按钮点击，但在我们在Nova公司使用的.NET版本上不起作用。

由于单元测试现在没有模拟面板，窗口飞过屏幕或在测试中等待输入时被卡住的可能性仍然存在。

最后，该解决方案在设计器生成的UI代码方面可能有更多问题，因为我们需要添加更多代码行，以将小部件暴露至外部世界。

GUI单元测试的其他可能解决方案：

我们可以利用GUI单元测试框架。这些是让你对小部件进行程序读写的框架，使用代码如`button1 = guiTesterUtility.find(“button1”)`这种方法的缺点是它们经常要用到字符串，在对小部件进行改名/重构时更容易出错，例如将“button1”改名为“CounterButton”。

将一个小部件工厂作为参数发送给面板。我们可以将所有对面板内新小部件的调用替换为对从外部发送的工厂的调用。因此，我们将得到

```
public void createComponents-
(WidgetFactory widgetFactory) { Button b1 =
widgetFactory.createButton(button1ID);
```

而不是

```
public void createComponents() { Button
b1 = new Jbutton();
```

生产中，工厂将创建实际JButton、JPanel等

```
Component contents = app.
createComponents( realWidgetFactory);
```

在测试中，它将创建具有Ids的模拟小部件。

```
Component contents = app.
createComponents( testingWidgetFactory);
```

然后，单元测试将能够通过其Ids引用这些按钮，以强制点击它们或读取其值。然而，Swing等图形框架要模拟许多小部件和功能，这可能需要做大量的工作。该解决方案还非常难于与图形设计器IDE进行集成，后者通常仅生成Button b1 = new JButton();等代码。

将GUI测试留给QA/自动化工具。这一最终选择似乎很滑稽，但实际上经常会出现。许多开发工程师做出明智选择，将精力集中于对逻辑的单元测试。

结论

GUI模块的单元测试是一项艰巨的任务。访问和编译控件的图形结果非常困难。我们有多种方法来解决这一问题，但每种方法都有一定的缺点。在我们的体验中，我们发现第一种解决方案，即将UI分成超薄的UI创建层和UI逻以色列知名辑层，是最佳折中方案。

Gonen Israeli

任职于以色列知名培训机构John Bryce Training公司，是软件开发以及开发方法论领域的一位专家培训师。曾任职于摩托罗拉半导体和IBM公司，并拥有丰富的项目开发经验。

责任编辑：高松（gaosong@csdn.net）

Redis源代码分析(上)

文 / 阮若夷

Redis是一个开源的Key-Value的内存数据库，以支持丰富的数据结构而著称，支持主从复制、持久化等高可用特性，可以和程序无缝地结合，本文从分析Redis的源代码入手，帮助大家了解这一款数据库的工作机理。

Redis是一个开源的Key-Value的内存数据库，与它类似的Key-Value数据库有memcached、Tokyo Cabinet等。Redis以支持丰富的数据结构而著称，还支持主从复制、持久化等高可用特性，可以和程序无缝地结合，而无须像使用关系型数据库一样需要ORM转换成关系型，或者像memcached一样只能使用简单的Key-Value。

memcached使用libevent这个已经不那么轻量级的网络事件库，而Redis本身不依赖任何第三方的函数库，无论是网络事件、散列表，数据结构都是自己实现的，全部代码只有2万行，算是一个中小型的项目，代码清晰，阅读起来非常流畅，甚至都无须Debug调试来辅助理解。

本文将分析Redis源代码，版本为2.2.2。下载源代码解压redis.tar.gz包后，进入redis目录，从src/redis.c的主函数开始我们的代码旅行。推荐读者使用cscope，这样可以很方便从函数之间跳转。

redis-server的启动过程

先从redis-server启动说起，自main函数开始遍历各个关键函数，了解一下Redis的主框架（如图1所示）。

首先initServerConfig函数会设置一些默认的参数，比如监听端口为6379，默认的DB个数为16等。PopulateCommandTable会把命令和函数数组转化成hash table结构，这个后文会详细描述。如果启动参数里有redis.conf，LoadServerConfig还会读入redis.conf里的参数，覆盖默认值。

initServer会给redisServer这个数据结构做初始化，申请各自成员的空间，其中有些是list结构，有些是dict结构。此后会添加一个时间事件，其函数为serverCron，并且每100ms执行一次，后文会详述这个函数的作用。接下来是启动监听，注册一个监听的文件事件，把accept行为注册到只读的监听文件描述符上。如果有激活aof功能，还会打开aof文件。接着会判断数据目录是否存在镜像文件或者aof文件，如果存在，redis会将数据载入到内存中。最后进入主循环。

主循环主要处理刚才注册的时间事件和文件事件。如何保证时间事件每100ms执行一次，又能即时处理网络交互的文件事件呢?

Redis处理得比较巧妙。先执行aeSearchNearestTimer，确定距离下次时间事件执行还有多少时间。假设第一次执行直到下次时间事件还有100ms，先执行文件事件，epool_wait的超时时间就设置为100ms，如果10ms后，有网络交互后经过一系列处理后消耗20ms，该次循环结束。aeSearchNearestTimer会再次计算距离下次时间事件的间隔为100-10-20=70ms，于是epoll_wait的超时时间为70ms，70ms之内如果没有处理文件事件，则执行时间事件。这样既保证了即时处理文件事件，又能在文件事件处理完毕后，按时处理时间事件。

时间事件serverCron会处理很多函数，例如定时打出日志展现Redis目前的状况，查看是否

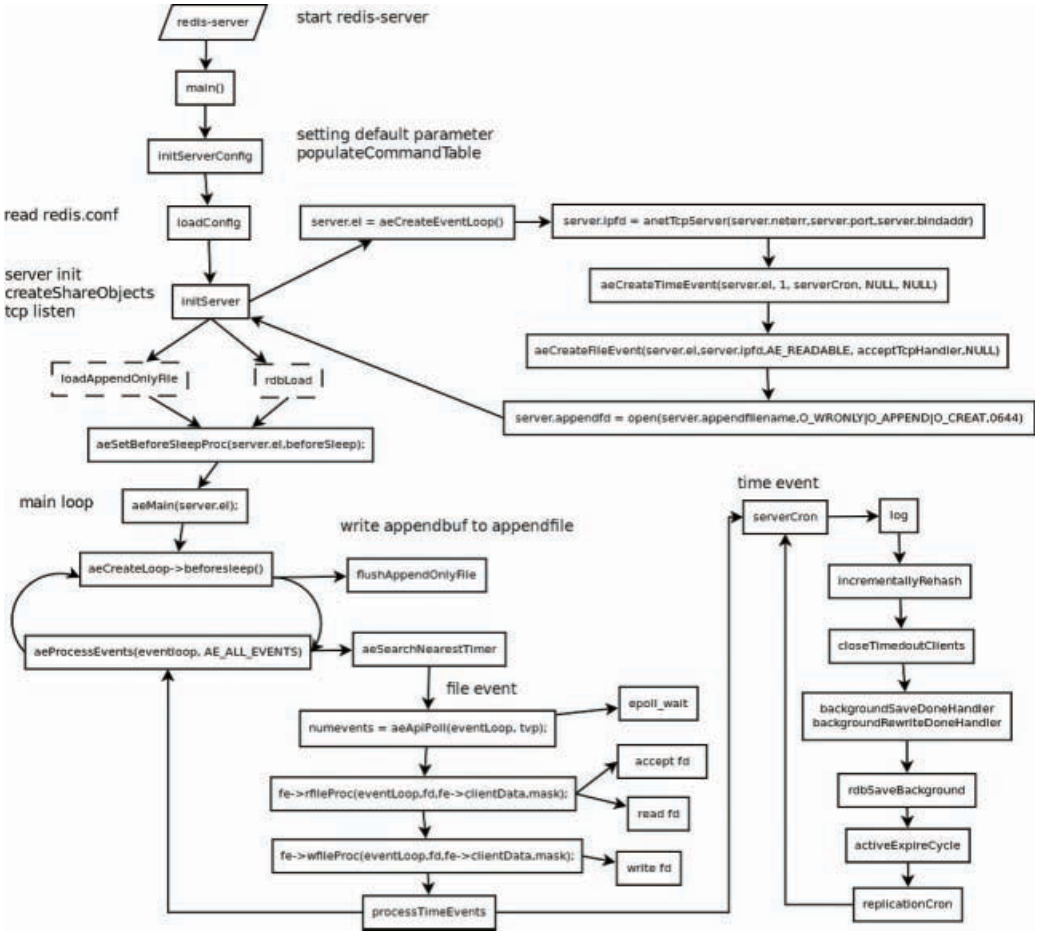


图1 redis启动示意图

需要rehash来迁移keys到新的bucket，这个后文会详细讲。

hash table实现

Key-Value数据库的KV查询的实现有很多种，比如功能全面的btree，而Redis的作者选择了简单的hash来实现，使用hash就意味着无法使用范围查询等功能，但择以更好的hash函数可达到更快的速度，而且代码实现更简单。

在Redis里hash无处不在，全局的Key-Value查询、内部的hash数据结构、命令与函数指针的关系都是使用hash。Hash的实现在src/dict.c、src/dict.h里。

dict为hash table的主结构体（如图2所示），dictht是为rehash而存在的中间数据结构，bucket就是hash算法里的桶，而dictEntry就为每个key-value结构体。dictht ht[2]指向2个dictht。存在2个ht的目的是为了在rehash时，可以平滑迁移bucket里的数据，而不像client的dict要把老的hash table里的一次性全部数据迁移到新的hash table，这在造成一个密集型的操作，在业务高峰期不可取。

每次Key-Value查询过程就是把要查询的key经过hash函数执行后的值与dictht->sizemask求位与，这样就获得一个大于等于0小于等于sizemask的值，定位到了bucket数组的位置。bucket数组的元素是dictEntry的指针。而dictEntry包含next指针，发生hash conflict时，直接以链表的形式加到链表的头部，所以查询是一个O(N)的操作，需要遍历dictEntry链表，而插入只插入到链表的头部，只是一个O（1）的时间复杂度。dictht->used表示这个hash table里已经插入的key的个数，也就是dictEntry的个

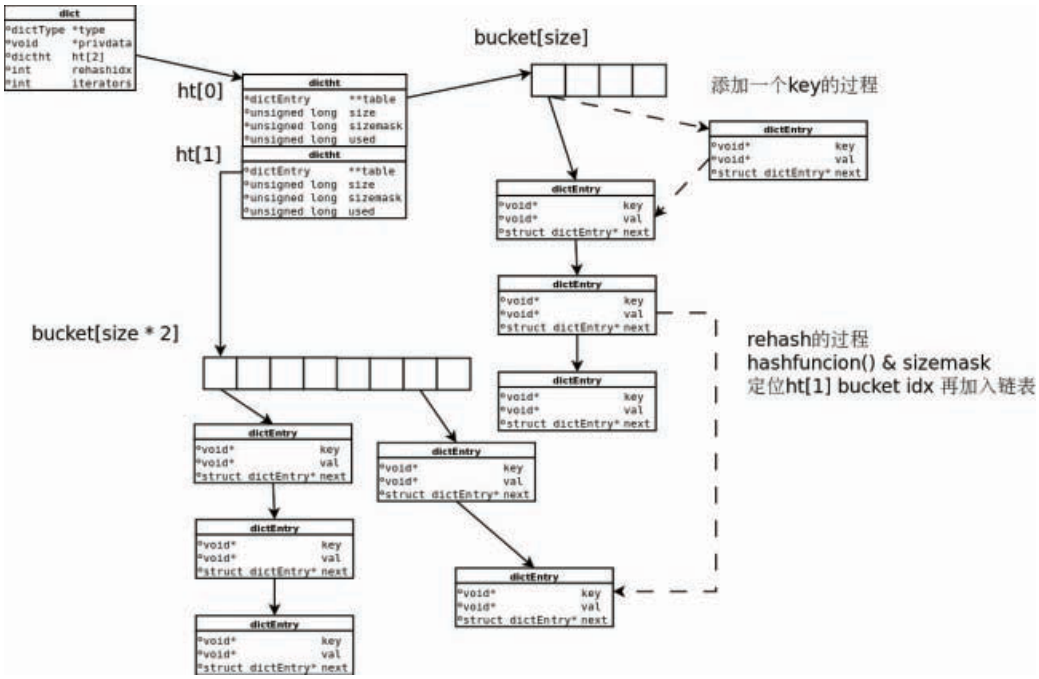


图2 redis-dict示意图

数，每次dictAdd成功会+1，dictDel成功会-1。

随着key不断添加，如果保持bucket数组大小不变，则每个bucket元素的单链表越来越长，查找、删除效率越来越低。

当dict->used/dict->size >= dict_force_resize_ratio（默认是5）时，就认为链表较长了，于是就有了expand和rehash，创建一个新的hash table（ht[1]），expand ht[1]的bucket数组长度为ht[0]上的两倍，rehash会把ht[0]上所有的key移动到ht[1]上。随着bucket数量增多，每个dictEntry链表的长度就缩短了。hash查找时，O（1）不会因为bucket数组大小改变而变化，而遍历链表从O（N)变为O（N/2）的时间复杂度。

rehash并不是一次性迁移所有的key，而是随着dictAdd、dictFind函数的执行过程，调度_dictRehashStep函数，一次一个将bucket下的key从ht[0]迁移到ht[1]。dict->rehashidx决定哪个bucket需要被迁移。当前bucket下的key都被迁移后，dict->rehashidx++，然后迁移下一个bucket，直到所有的bucket下的key被迁走。

除了dict_add、dict_find出发rehash，另外redis运行过程中，会调用dictRehash-Milliseconds函数，一次rehash 100个bucket，直到消耗了1秒才结束rehash，这样即使没有发生查询行为也会进行rehash的迁移。

rehash运行的具体过程如下：遍历dict->rehashidx对应的bucket下dictEntry链表的每个key，对key进行hash函数运算后与ht[1]->sizemask求位与，确定ht[1]的新bucket位置，然后加入到dictEntry链表里，然后ht[0].used--，ht[1].used++。当ht[0].used=0，释放ht[0]的table，再赋值ht[0]= ht[1]。

在rehash的过程中，如果有新的key加入，直接加到ht[1]。如果key的查找，会先查ht[0]再查询ht[1]。如果key的删除，在ht[0]找到后则删除返回，否则继续到ht[1]里寻找。在rehash的过程中，不会再检测是否需要expand。由于ht[1]是ht[0]size的两倍，每次dictAdd时都会迁移一个bucket，所以不会出现ht[1]溢出了，而ht[0]还有数据的状况。

使用hash table的地方

举个最常见使用的例子，RedisServer存储全局key-value的RedisDB。

每个Key-Value的数据都会存储在RedisDB这个结构里，而RedisDB就是一个hash table。从图3上我们可以看出key为“hello”，value为

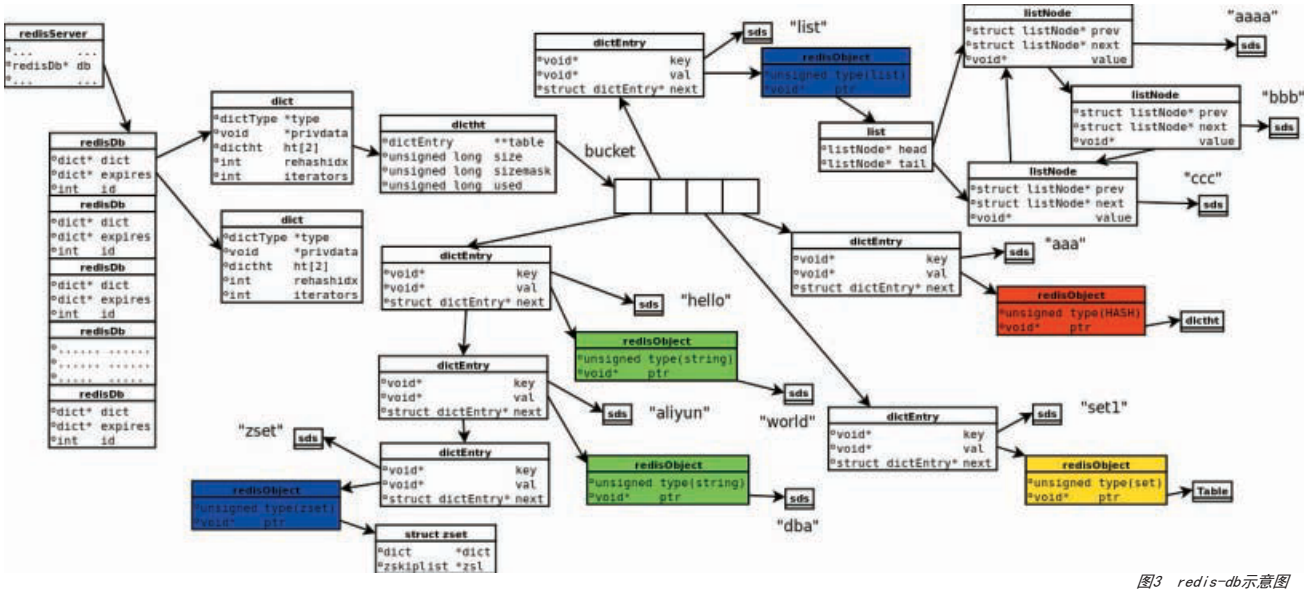


图3 redis-db示意图

“world”的存储格式。key为“list”，value为一个字符串链表（["aaa","bbb","ccc"]）的存储型式，另外的set、hash、zset就不一一表示了。

另外常见的命令（例如get、set等）会调用的函数指针，这个数据结构也是以hash table的形式存储的。每次客户端输入“set aa bb”等数据时，解析得到字符串“set”后，会根据“set”作为一个key，查找到value；一个函数指针（setCommand），再把“aa”、“bb”作为参数传给这个函数。这个hash table存储在redisServer->command里，每次redis-server启动时，会对readonlyCommandTable这个数组进行加工（populateCommandTable）转化成redisServer->command这个hash table方便查询，而非遍历readonlyCommandTable查找要执行的函数。

一个redis命令的历险

我们通过客户端发出一个普通的“set a 1”的命令来遍历Redis，来遍历Redis几个关键函数，并熟悉协议处理的过程。

你可以通过telnet或者redis_cli、各种程序语言bind的lib库发送请求给RedisServer。前者以一种裸协议的请求发送到服务端，而后两者会对请求进行协议组装，帮助更好的解析（常见的固定位置放置请求长度和协议类型）。我们来看看请求和回复的协议格式。

请求协议：

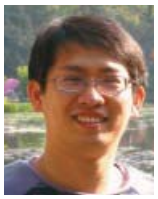
```
*参数的个数\r\n
$第一个参数的长度\r\n
第一个参数\r\n
...
$第N个参数的长度\r\n
第N个参数\r\n
```

例如请求“get a”，经过协议组装后的请求为：

```
2\r\n
$3\r\n
get\r\n
$1\r\n
a\r\n
```

以上过程可以帮助我们了解到遍历关键函数的Redis命令的处理过程。（未完待续）

-



阮若夷

目前就职于阿里云计算运维部，是阿里云关系型数据库云服务（RDS）和鹰眼监控的开发者。

责任编辑：高松（gaosong@csdn.net）

技术雷达 Technology Radar

ThoughtWorks 技术咨询委员会 2011年7月

技术

持续交付强调最大程度的自动化——包括**基础设施即代码**、环境管理和部署自动化，从而保证系统随时准备好输出产品。这还关系到缩短反馈回路，以及始终坚持及时完成工作。持续交付和**持续部署**不同，持续部署意味着每一个改动都部署到产品，持续交付让你掌管整个产品环境。企业可以选择部署什么、何时部署。

改善开发和操作之间的互动关系，可以让交付和生产系统更有效、更稳定和更易维护。建立DevOps文化需要注意团队组织、工作实践、汇报程序和激励措施，以形成一种共同责任，促进更加快速和更加安全的交付。

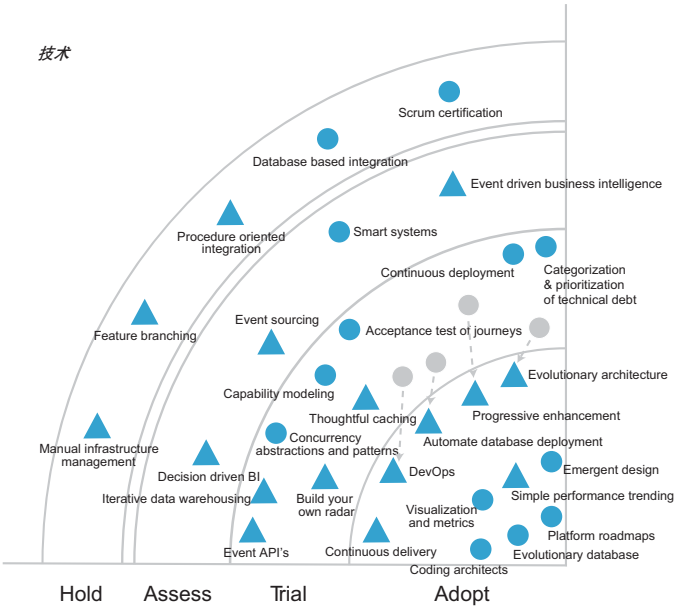
建议抛弃需要前期投入、重量级的传统企业架构设计，转而采用**演化式架构**。这种企业架构的好处是，避免了因为试图精确预测未来而造成的问题。演化式架构具有可适应性，不需要再预先设想该如何复用组件；通过合适的抽象、数据库迁移、测试套件、持续集成和重构，问题发生时自然可以迎刃而解。要提前认识到系统中的主要技术需求，以保证在后续的设计和实现中能够对它们处置得当。

REST API已经成为了产业标准。良好的REST API提供了一种简洁轻量的方法，来实现自定义和集成。但很多情况下，要想实现快速、高质量的集成，必须知道某些事发生的时刻。这时，我们可以考虑建立**事件API**，将它和REST API配合使用，能够促成简单的工作流程、通知和同步集成。这些集成所需要的代码通常不过20~30行。事件API的形式通常是“Web钩子”或回调机制，但基于轮询的Atom风格事件API也是可行的。Atom事件API能够低成本地扩展并且让客户端能够确保交付完成。

SOA的目的之一是通过可读的业务文档，而非编程参数，达到分离服务的目的。但实施SOA时，很多企业使用了Web服务来展示后端系统的底层编程模型。**面向过程的集成**只不过是经由不同协议实现的远程过程调用，其后果是更多的层增加了复杂性却没有改进业务的灵活性。为了避免这一点，在实施SOA时，首先需要理解这些服务的业务意义，然后实施独立于旧有系统的可读契约。过于频繁的缓存是一种事后补救措施，用一刀切的方法和共同的缓存存活时间来解决性能问题。这造成了一些问题，也引出了一些变通办法。我们认为**精心安排的缓存**应该是在项目初期就处理的事情，不要仅仅把它当成最后一刻的性能修复。

定期做一些能够检查系统关键部分的极其简单的性能测试，就足以很好地追踪趋势，这样我们看到性能的变化时，就能及时反应。当天或者隔夜对编译版本做这些测试，并且把结果作图来完成简单的**性能趋势分析**。在真正的典型环境中进行复杂性能测试仍然有用，但不能依靠它们来了解代码性能的变化。

如果把业务变化的速度看做需求变化的指示器，那么预先数据库设计的年代已经过去了。取而代之的，项目开始采用**演化式数据库技术**，在项目进展过程中，根据新需求不断地改变数据库方案。数据库变动的部署也应该自动化，好让依赖于这些变动的应用发布不必等待数据库变动的手动部署。**数据库自动部署**可以应用和数据库变动的自动部署。演



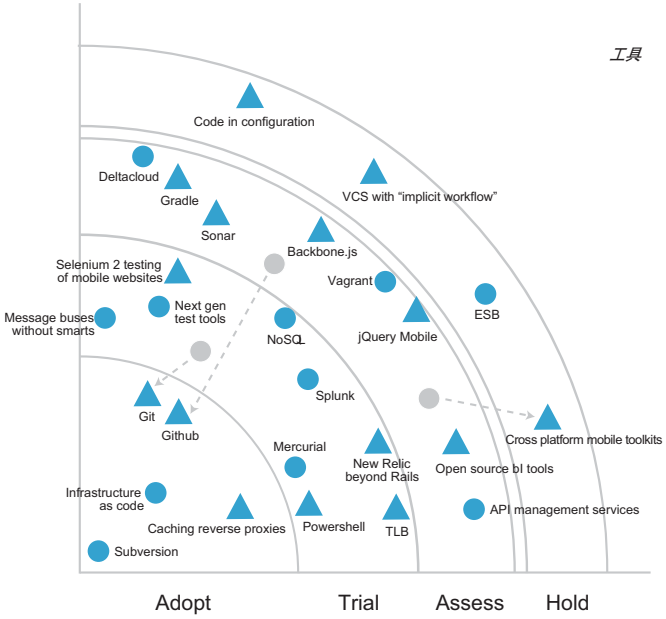
化式数据库和数据库自动部署为高生产率的团队提供了一种保证持续交付的方法。

尽管自动化技术有了很大进展，很多人还是退回去采用**手动基础设施管理**。我们经常会见到防火墙和负载平衡器的手动配置造成的问题，特别是DBA在产品数据库中剪切和粘贴SQL代码带来的问题。这些操作，即使不完全自动化，至少也应该写成脚本，重复执行。令人遗憾的是，我们还会看到开发团队使用**特性分支**来隔离工作和延缓集成。特性分支常常会给稍后的集成工作带来很大的麻烦和很多的不可预测性，但更主要的是，它妨碍了维护高质量软件所必需的持续设计改进。我们建议用持续集成和抽象分支来代替特性分支。

最近，**渐进增强**在移动应用上的使用非常有效，表明了这种Web设计策略的通用性。我们建议采用这种策略来保持代码的简洁，同时为用户带来适用于他们设备的最佳体验。

事件溯源是一种持久性的数据记录，主要内容是引起变化的所有事件的日志。传统的数据库状态可以通过对事件记录的再处理完整地重现。事件溯源的好处包括强审计、历史状态生成和可以重放事件以便调试和分析。实现数据仓库和商务智能的传统方法是沿水平层级自下而上，从整个企业的数据源中收集并净化数据，然后整合到一个总的数据仓库中，以供生成报告。有些人现在采用另一种方法，从真实的输出（业务决策）开始，然后抽取支持该决策所需的要素。**决策驱动的商务智能**允许一种实现商务智能的渐进方法，促进对商务智能的终极客户——决策制定者的快速反馈。

使用迭代技术实现数据仓库有很多好处。**迭代式数据仓库**技术允许数据仓库的终端用户决定他们需要什么样的报告，然后由ETL开发者和数据建模师来实现这些功能，从而避免了在数据建模和ETL工作中，不能为企业马上带来价值所造成的时间浪费。**数据可视化**在制定商务和IT决策时很有效，可以有效地利用实时数据。这些可视化包括某个时间点上的数据和数据随时间的变化趋势。



工具

DevOps运动还在继续扩大，开发者和操作人员紧密合作，解决“软件最后一公里”问题。**基础设施即代码**，这项技术像对待代码一样对待基础设施配置。除Web服务器、应用服务器和应用配置之外，我们也希望同样看待网络配置。网络交换机、防火墙和负载均衡配置这些基础设施可以被看做代码，甚至可以在运行中更改。

测量软件内在品质仍然是个神话，尽管有很多源代码指标存在多年。这些指标的问题在于它们通常只关注品质的一个方面。必须参阅很多指标，才能对代码总体品质做一个结论。Sonar是一个用来检查、追踪和可视化这些指标的整合工具。它不仅仅将指标结合在一起，而且还将它们和历史测量结合，让我们对代码库的内在品质有更深入的了解。

很多机构想要尽量减小IT生产环境中的变化。这经常会造成行为的反模式。一个例子是过度使用**配置中的代码**来影响生产系统的行为。在配置文件中而非代码中修改，而配置文件却无法进行与应用程序同等级的测试。

如果测试套件越来越慢，而且已经认定这不会对应用造成大问题，那么首先请将测试速度加快，然后试试并行化。**测试负载均衡器**（TLB）是并行测试执行方面的一个大进展。它使用智能算法和历史测试数据来优化工作量分布，减少延迟时间，免除了低效的手动任务分发。更进一步，它智能地调整测试，比如说首先进行上次执行中失败的测试，以更快速地获得反馈。并行执行可在计算机网格间进行，也可在一台计算机的多个进程间进行。JUnit、RSpec、Test::Unit、Twist和Cucumber目前都可以支持，对NUnit的支持还在开发中。引入**缓存反向代理**后，应用设计更简单，而且对于基础设施的故障也更有弹性。在应用和Web服务之间放置缓存反向代理降低了服务故障的风险，也改进了整个系统的性能。

尽管JavaScript在如今的软件开发中扮演了越来越重要的角色，要用简洁的结构组织它仍然是件麻烦事。Backbone.js是一个为JavaScript重度应用提供MVC框架的库。它允许开发人员以更易管理和易测试的方式书写

JavaScript代码。

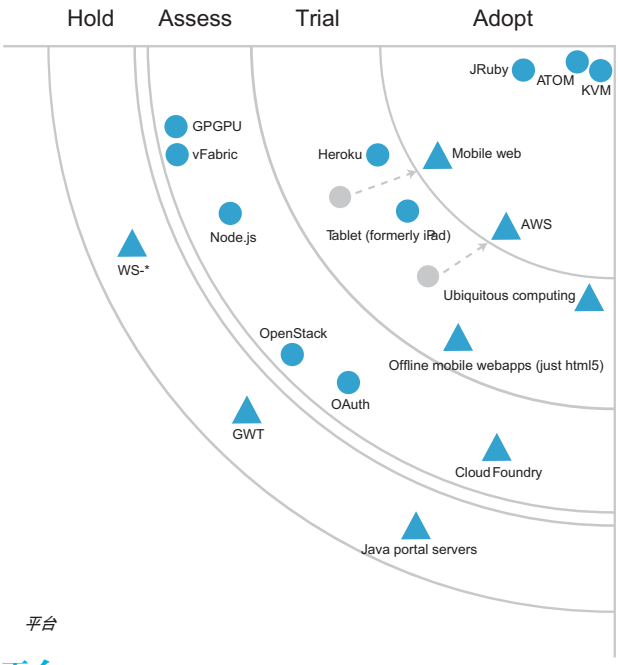
Git实现了结构良好、高执行性能的分布式版本控制。它的功能很强大，可以实现其他工具中无法实现的敏捷工程工作流程。Git的普及背后有GitHub的支持。GitHub整合了公开的和私有的Git资源、社交网络，还有很多其他的创新工具和方法。有些工具是为了实现和促进不同的工作方式。不幸的是，还有一些工具恰恰相反，它们对用户有着低信任度，并且要求遵循预先定义的过程。 ClearCase和TFS就是两个例子。这些有**“隐含工作流程”的版本控制系统**不适于现代的敏捷软件开发。由于过分强调某些事情（比如说登入），这些工具只会碍事和降低生产力。

ThoughtWorks在两个移动网站项目中使用了**jQuery Mobile**。一个项目中，这个库在处理设备差异和旧浏览器中的优雅降级方面非常有用。但在另一个项目中，这个工具并不那么有用，它会强制项目以某种特定的方式进行，这种方式并不十分适合当下的应用。开发移动应用时，我们可以使用**Selenium 2移动测试**在iOS、Android和黑莓上进行同样的验收测试。它可以工作在模拟器、仿真器和物理设备上。我们已经成功地使用这种方法在为所有这3个平台制作的软件产品上。黑莓的驱动还在测试阶段，但它已经足够稳定。

那些声称能在Windows、Linux和OS X上创造无缝用户体验的工具并不言副其实，不得不在一个或多个操作系统上牺牲体验。跨平台移动工具包在实际项目中取得了不同程度的成功。我们发现了一些问题，包括必须为每个平台分别创建一个工程和调用特定的本地UI部件来搞定事情。

NoSQL技术已经一天天成熟。MongoDB、Riak、Neo4J、Cassandra还有很多其他技术都在帮助扩大 NoSQL的领地。

开源商务智能工具，比如说Pentaho、JasperSoft、CloverETL、Talend、BIRT和SpagoBI已能与商业专有工具相抗衡。我们建议尝试并评估它们。在开发和生产中，我们用New Relic对产品和开发Ruby on Rails系统进行性能监控。快速安装和综合报告的结合在性能故障诊断中非常有用。我们从New Relic对Java和.NET系统的监控服务中也看到了很好的效果。Powershell是管理Windows服务器和应用的重要工具。Powershell被集成在了Windows 2008和Windows 7中，允许Unix式的脚本语言和跨服务器群的自动化。脚本可以在远程计算机上执行，一条命令就能同时管理几百台计算机。Powershell脚本可以部署和配置应用程序和操作系统组件，而且可以通过编写.NET commandlet进行扩展。Gradle允许你与已有的Maven软件库互动，但通过一种简洁的领域特定语言为你的编译版本提供可脚本化。



平台

平台

我们将**移动Web提升为采纳状态**。我们认为这是为移动设备提供网络内容的一条正确途径。HTML5的一些特性允许使用客户端的JavaScript在浏览器中控制和存储离线数据。这些特性允许创建跨平台的**离线移动Web应用**，而不需要像以前那样安装应用程序。HTML5标准还没有最终定案，但对这些离线特性的支持已经就位，可以应用在基于Webkit的浏览器上。平板设备提供了一种新的计算模式。新一代的**平板电脑**展示了新的互动方式的潜能，我们认为它还会带来更多的创新。

普适计算是一个难理解的术语，它包含了很多不同的思想。目前我们发现的有趣和令人兴奋的一点是，消费类移动设备和专门移动设备都越来越多地基于像Android和iOS这样的商业操作系统。这意味着，在很多情况下，机构可以自己开发软件，不需要花大钱雇用专业人才就能进行新应用的创新。硬件价格的下降也让这个领域更容易进入，特别是有丰富外围设备的支持，比如说付款读卡器、密码键盘设备和高品质条码扫描器等开始支持Android和iOS设备。在它们和这些消费类设备已有的特性结合后，将开辟出了新的工作方式。

OAuth是一种Web友好的轻量级标准，用来批准授权，允许用户在网络服务之间分享私人资源。它使用简单，避免了密码的激增，可以给予某服务最小的授权。如果你考虑以一种轻量级、网络友好的方式展示你的应用数据，强烈建议

使用OAuth作为授权标准。

Charles Nutter和JRuby团队继续以惊人的速度改进JRuby，包括DB适配器、gem管理和现代Rails部署。Rails3结合JRuby是很棒的平台。在企业中，没有理由不使用Ruby。

Amazon继续发展AWS云服务，比如说它的关系数据库服务，使得设计和部署基于云的应用更加简单。但不是所有AWS的功能都像EC2和S3那么成熟，在需要弹性或按需计算的情况下，我们推荐采用AWS的服务。

Cloud Foundry是开源的PaaS，可部署在自有的数据库上，也可交给VMware托管。Cloud Foundry支持Java/Spring应用、Rails、Sinatra、Grails和node.js。附加的服务包括MongoDB、MySQL和Redis。对Scala和Lift的支持，活跃了该平台的开发。但它与vFabric的关系还有待观察。

Heroku是非常优美简洁的PaaS。尽管从Ruby on Rails平台发展起来，但现在已经支持很多的语言和Web框架，最近一个是Clojure。Heroku使用标准栈和可通过简单的Git push部署应用。被Salesforce收购后质量未见下降。

持续出现交付问题的一个原因是使用了Java Portal Server包。这些问题在开源和商业门户平台中都会出现。复杂和不方便的编程模型，很难进行自动部署、数据整合和测试，影响了这些平台，让它们无法达到许诺的生产力。尽管产品展示很吸引人，但门户产品的基本功能通常不能很好地适应真实的网络应用。而特别广而告之的功能，比如说单点登录或搜索已经由现有的有针对性的企业资产提供。

GWT的架构选择很糟糕，但实现还不错。 GWT试图隐藏Web平台的很多细节，通过用Java创建桌面隐喻，并生成JavaScript代码来实现它们。首先，在很多方面，JavaScript比Java更强大更具有表现力，所以我们怀疑它们之间的关系搞反了。其次不可能隐藏复杂的抽象差异，就像从事件驱动的桌面到无状态的Web，最后总会有令人头疼的遗漏的抽象跳出来。最后，它和很多精心制作的框架有着同样的缺点，用它们建立简洁整齐的应用可以很快很容易，建立复杂但没有支持功能的应用可能但是困难，建立一个大应用所需要的复杂程度是不可能的，或者因为太困难而不可行。

以前我们的建议是，在Basic Profile之外轻易不要使用WS-*。随着更简单的Web平台技术（如REST和OAuth）的发展和被接受，以及对WS-*已有问题的了解，现在可以使用它，不过要小心一些。

语言

HTML5是一个演进中的标准，很多部分已经达到了可以安全应用的阶段，可用来创造在线或离线的移动Web应用。根据项目实践，我们认为HTML5已经准备好在移动Web应用中采用。随着标准的继续演进，因为它支持跨平台的独特优势，预期HTML5会逐渐变成本地应用的可靠替代。

Clojure是动态的函数语言，可以应用在Java虚拟机上。它吸收了很多现代编程观念，包括惰性求值和高级的并行抽象。Clojure催生了充满活力的程序员社区，提供了丰富的框架和工具。其中一个例子是Midje，创新的单元测试和Mock框架。

JavaScript是一种强大、应用广泛的编程语言，有着麻烦而且容易出错的语法。Coffeescript修改了JavaScript的很多缺点，通过干净简单的语法生成可读的JavaScript。为什么我们不喜欢GWT，却拥护Coffeescript，这可能会让一些读者感到困惑，因为从表面上看来，这两者很相似：都是用来生成JavaScript的工具。但它们的抽象程度不同。GWT有详细的组件模型，用来掩盖底层语言（JavaScript）和平台（Web）中相关的细节。Coffeescript让编写正确的JavaScript变得简单，避免了JavaScript中病态的默认“特性”，而且不会用层把开发者和平台隔离开。

令我们吃惊的一点是，在2011年，我们还能找到用**存储过程中实现业务逻辑**的新系统。通常，用来执行存储过程的编程语言缺乏表现性，不容易测试，不容易进行简洁的模块化设计。只有在被证实存在性能问题的特殊情况下，才能考虑在数据库引擎中执行存储过程。

新书上架

本月新书

0day安全：软件漏洞分析技术

主编：王清

本书系统全面介绍Windows平台软件缓冲区溢出漏洞的知识，囊括了Windows平台高级溢出技巧、手机平台的溢出基础、内核攻防、漏洞挖掘与安全测试、大量的0day分析案例等。

iPhone应用程序开发攻略

编著：王志刚 等

介绍了使用Objective-C 2.0开发iPhone应用程序的基础知识，涵盖各种开发工具的操作技巧、框架内部的工作原理、软件的调式技巧、后期制作等基础知识。

C#程序设计语言

作者：Anders Hejlsberg等

译者：陈宝国 黄俊莲 马燕新

C#语言结合了快速应用开发语言的高效和C/C++语言的强大。本书全部内容更新到C# 4.0版，提供了C# 4.0语言的完整规范、参考资料、范例代码和来自12位卓越的C#大师的详细注解。

ExtJS Web应用程序开发指南

作者：卫军 夏慧军 孟腊春

ExtJS前身是YUI，经发展与改进，现成为最完整成熟的一套构建RIA Web应用的JavaScript基础库。ExtJS已成为开发具有良好用户体验的Web应用的完美选择。

浮现式设计：专业软件开发的演进本质

作者：Scott L.Bain

译者：赵俐 华洁

JOLT大奖得主。作者Bain向我们介绍了一次只做一个步骤的自然而生的设计的原则和实践。

推荐图书

松本行弘的程序世界

作者：松本行弘

译者：柳德燕 李黎明 夏倩 张文旭

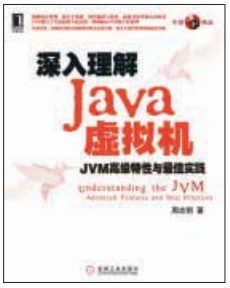
出版社：人民邮电出版社

在流行的编程语言中，Ruby比较另类，这是因为大多数编程语言的首要着眼点在于为解决特定的问题领域而设计语言，而Ruby的首要着眼点在于“人性化”，让程序员充分享受编程的乐趣。作者松本行弘是一位性格平和、对生活充满热爱的人，他信奉code for fun的宗旨，即编程语言不应该是冷冰冰地给机器阅读和执行的指令，而应该是让程序员编程的工作过程变成一种充满乐趣和享受的过程。而且，松本先生发明Ruby语言也是因为他对创造一种人性化的面向对象脚本语言的热爱。

当然，Ruby并非只在非主流程序员社区中流行，随着全球IT产业向云计算时代的发展，Ruby也发挥着越来越大的作用。著名的SaaS厂商salesforce在2010年底以2.1亿美元收购了PaaS厂商Heroku，并且在2011年7月聘请松本行弘担任Heroku首席架构师，开拓Ruby在云计算领域的应用。Heroku本身就是一个完全用Ruby架构的PaaS平台，同样支持Ruby的PssS厂商还有EngineYard、VMware等。这些云计算厂商的努力，使Ruby必然在未来得到越来越广泛的应用。

这本书实际上是松本行弘从一个编程语言设计者的角度去看待各种各样的流行编程语言，包括它们有哪些特点，以及Ruby编程语言是如何取舍的。Ruby编程语言的设计本身大量地参考了一个更古老而著名的面向对象编程的开山之作Smalltalk，而且从函数式编程语言鼻祖LISP“偷师”了不少好东西。

程序员社区有个著名的说法：任何现代编程语言都脱胎于Smalltalk和LISP，都可以从这两个编程语言身上找到似曾相识的特性，自Smalltalk和LISP诞生以来，编程语言领域大势已定。因此集这两种编程语言很多特点于一身的Ruby语言很值得编程爱好者去学习，而看看Ruby设计师是怎么设计Ruby语言，则可以让你高屋建瓴地理解一些主流的编程语言。



深入理解Java虚拟机

JVM高级特性与最佳实践

作者：周志明

出版社：机械工业出版社

Java程序是如何运行的？Java虚拟机在其中扮演了怎样的角色？如何让Java程序具有更高的并发性？许多程序员都会有诸如此类的疑问。

本书是一本从实际应用的角度讲解Java虚拟机的著作，全书共包含五个部分。第一部分从宏观

的角度介绍了整个Java技术体系，这对理解后面的内容很有帮助；第二部分讲解了JVM的自动内存管理，包括虚拟机内存区域的划分原理和各种内存溢出异常产生的原因，常见的垃圾收集算法以及垃圾收集器的工作原理；第三部分分析了虚拟机的执行子系统，包括虚拟机的类创建机制以及类加载器的工作原理和它对虚拟机的意义，虚拟机字节码的执行引擎以及它在实行代码时涉及的内存结构；第四部分讲解了程序的编译与代码的优化，讲解了虚拟机的热点探测方法、HotSpot的即时编译器、编译触发条件，以及如何从虚拟机外部观察和分析JIT编译的数据和结果；第五部分探讨了Java实现高效并发的原理，包括JVM内存模型的结构和操作；原子性、可见性和有序性在Java内存模型中的体现，虚拟机实现高效并发所做的一系列锁优化措施。

互联网之达芬奇密码
浪潮揭秘：与中国五亿网民互为影响的互联网DNA
作者：Mull He
出版社：电子工业出版社

互联网曾改变了我们，现在，由我们改变互联网。
1995年中国电信开通两个接入Internet节点，网景Netscape上市，打开了人们关于互联网公司的种种商业想象，也开启了中国互联网的商业化进程。十多年来，日益丰富和层出不穷的互联网应用和服务让中国“用户”们的生活和工作方式发生了翻天覆地的变化。
自互联网行业进入中国开始商业化进程以来，各种服务都是应用户需求而产生的，不满足或者不能持续满足用户需求的产品或企业，都非常难“生存”下来。每两三年或者三五年，不同时间阶段都会造就出不同领域的成功者，也都会带来新的工作和生活方式。这一切都是由一个最根本的因素在推动——用户需求，以及用户需求的不断发展和变革。到目前为止，几乎所有成功的互联网商业模式都是因顺势而为赢得用户需求而获胜的。“先用户而后盈利”几乎成为了互联网模式发展的规律。回过头来仔细地分析互联网发展史，我们会发现每一个阶段的代表性模式都表现出紧贴用户需求的明显特征，如笔者几年前所提到的：URL并不只是网页地址（Universal Resource Locator），而是我们每个人、每个用户在互联网上要去的最终地址。在某种意义上，我们可以将这个地址理解为User Requirement Last，这也正是与中国4亿多网民互为影响的互联网DNA。
本书特别之处在于用分章节的形式讲述了中国互联网十多年发展历程中各个阶段的真实用户需求、企业发展、产业历程。在对应领域和时代背景的基础上，以独到的理论来分析中国互联网每一次变革与用户需求演进之间的潜在关系。读者可以从本书的阐述中体会到互联网与我们的生活、学习、工作、娱乐方式之间互为影响和互动进化所产生的颠覆性的革命浪潮和机遇，可以分享到中国互联网由用户需求而推动的进化过程，进而从互联网发展与用户需求演变的规律中，理解、掌握和运用这种规律来追求积极和未来成功之道。

全球排行榜

Amazon

01 Network+ Guide to Networks (Networking (Course Technology))
02 Head First HTML with CSS & XHTML
03 Getting to Know ArcGIS Desktop
04 GIS Tutorial 1: Basic Workbook
05 Introduction to Algorithms
06 Network Fundamentals, CCNA Exploration Companion Guide
07 Visualize This: The FlowingData Guide to Design, Visualization, and Statistics
08 Head First Java, 2nd Edition
09 C Programming Language (2nd Edition)
10 HTML, XHTML, and CSS, Sixth Edition

天珑书局（中国台湾）

01 前进Android Market！Google Android SDK实战演练
02 深入浅出Android系统移植与开发测试
03 Google Android SDK开发范例大全
04 大话数据结构
05 探索iPhone 4程序开发实战
06 超图解Excel VBA基础讲座
07 网页接口设计模式
08 HTML5：建置与执行
09 培养与锻炼程序设计的逻辑
10 深入浅出Java程序设计

第二书店

01 智能Web算法
02 你必须知道的.NET
03 大规模Web服务开发技术
04 浪潮之巅
05 深入理解Java虚拟机：JVM高级特性与最佳实践
06 Java编程思想
07 让Oracle跑得更快
08 项目经理应该知道的97件事
09 大话数据结构
10 软件研发之道：微软开发团队的经验法则

GEEK产品

LOMO spinner 360°

Lomography推出机械式相机，可拍摄360° 全景的Spinner 360° 。在拉下快门的同时，相机会跟着360° 旋转，一瞬间便可记录超过传统底片4倍的影像，包括拍摄者与背后的全景。

Ciclotte exercise bike

运动减肥也可以很时尚。这款室内健身自行车由意大利时尚设计师Roberto Cavalli设计，全车采用碳纤维、钢与玻璃制造，轻便、简洁，将简单性和实用性结合得恰到好处。

Qrobot

Qrobot是由腾讯和中科院合作推出的一款智能互联网机器人，硬件配置有摄像头、电容式触摸传感器、SRS环绕立体声解码器和高品质音箱。可以通过语音的方式交谈、下达指令，包括报天气、股指、新闻等等。内置可升级的管理软件。

I-MU共振音箱

I–MU系列音箱采用独特的共振原理让接触的硬质平面震动发声，并且能使声音穿透平面，在另一边也能听到同样的声音。此系列中根据中国生肖文化设计的几款音箱（I–Jerry和I–MO）还内置了MP3播放器，外型更为可爱。

Yill

德国设计师Werner Aisslinger为一家能源公司设计了这款苹果风格十足的移动电源。这个电源提供300W的电力供应，内部的锂钛电池可以通过太阳能充电，也能通过外接电源充电。体积虽然比较大，但是带有两个轮子可以很容易地放到家里的任何一个角落。

一切弯路都是直路

英特尔首席工程师吴甘沙访谈

记者 / 董世晓

作为从英特尔中国研究院走出的第一位首席工程师，吴甘沙的故事或许能给我们一些启迪。他相信无论做任何事情，只要你认真对待，即使走的是弯路，同样也能获得另外一种成功。

首席工程师的责、权、利

《程序员》：首先恭喜您成为英特尔的首席工程师，请介绍一下首席工程师的工作职责。

吴甘沙：根据我的理解，首席工程师不是一个具体的工作岗位，更多的是一种荣誉，或者是一个职称。英特尔共有8万员工，其中近5万人从事技术工作，因此就需要有一个系统，能够让每一个人都能在其中找到自己的定位以及努力的方向。所以我们创造了一个系统的职业阶梯，使新加入的人可以从基层开始，慢慢往上攀登，最终走到首席工程师、资深首席工程师、院士、资深院士等位置。

前两天我女儿唱她新学的《葡萄树》给我听，里面有一句歌词叫做“蜗牛背着重重的壳，一步一步往上爬”，让我深有感触。其实我们就像蜗牛，很可能永远爬不到这棵葡萄树的顶端，永远吃不到葡萄，但追求的过程却是人生的真谛所在。

当然，首席工程师也有其责、权、利。

“利”肯定是有的，一方面会加薪；另一方面，还能享有一些专属待遇。比如我们公司每年都会举办院士战略论坛，讨论公司最核心的技术以及未来的方向，首席工程师就有机会参加这个论坛。

“权”相对来说比较少。我们有40~50位院士，很多都是光杆司令，不见得会带很大的团队，更多的是在技术上做贡献。

关于“责”，我是从三个人对我的期望考虑的。第一个人是我母亲，她听说我成为首席工程师后，说的第一句话就是“盛名之下，其实难副”，一方面要我低调，另一方面首席工程师在一些特定的领域有特殊的贡献，在整个学术界、工业界有一定的名望，所以必须保持技术上的精益求精。第二个人是方之熙院长，他对我提出了战略领导力方面的期望，这包括两层含义：时间上，要看得足够远，要为公司找到合适的发展方向；空间上，不应只局限于自己的一亩三分地，在成为首席工程师后，必须得从整个研究院的角度看问题，更多地考虑和帮助研究院的发展。第三个人是我的院士导师，他说当你小有成就的时候，就需要考虑将更多的机会给手下的人，给他们发展的机会，帮助他们成长为下一个首席工程师，这是从人才培养的角度来说的。

吴甘沙：只要你是认真地对待，一切弯路都是直路

阴差阳错进入计算机行业

《程序员》：刚才提到《葡萄树》这首儿歌，比喻很形象，那您儿时的梦想是什么？

吴甘沙：我儿时梦想成为画家、作家或者音乐家。至于科学家，我丝毫没有想过。当时对科学的喜爱，更多的是看一些关于UFO的杂志。但事与愿违，就像《老男孩》里面唱的那样“梦想总是遥不可及”，我走上了技术之路。当然，一般来说，如果童年的梦想没有实现的话，就会从另外一些方面获得弥补。我夫人是中央工艺美术学院的，多多少少圆了我当画家的梦想；我给女儿专门买了一套很好的音响，希望她在音乐方面有所建树；我自己也在尝试写一些工作的感悟，希望有朝一日能够分享给大家。

《程序员》：在读大学之前，有没有一些印象深刻的经历？

吴甘沙：大学之前基本上是一帆风顺的，人往往就是这样，如果太顺利的话，在回想过去时，就找不出什么特别的闪光点。但临近高考，还是发生了一件影响我一生的事情。在高考之前的全省会考中，我成绩非常好，如果没记错的话，应该是全省第一。于是，我很骄傲，在高考之前的那段时间里玩得过火，但骄傲肯定是要付出代价的，结果高考成绩不是很理想，尽管考入了复旦大学，但没能被第一志愿“遗传工程”所录取，阴差阳错地读了计算机专业。虽然说我现在不后悔，但这段经历确实给了我深刻的教训。

这个挫折也让我懂得了另外一个道理——“只要你是认真地对待，一切弯路都是直路”，也就是说可能一开始走上了一条弯道，但如果沿着这条弯道好好走下去的话，同样也能够获得另外一种成功。

《程序员》：这就像海伦·凯勒的那句经典“上帝为你关闭了一扇门，就一定会为你打开一扇窗”。那您是从大学开始接触编程的吗？

吴甘沙：这点很惭愧，我在高中时摸过两次电脑，但都没来电。在得知将会读计算机专业后，我赶紧买了一本Basic编程的书。当时大学还需要军训40多天，于是我就揣着这本书，坐上了去大连的轮船。军训期间，忙里偷闲，我把这本书看完了，感觉略有小成，但基本都是在脑子里编程，连键盘都没摸过一次。一进大学我就傻眼了，学的是跟Basic一点关系都没有的Fortran。更有甚者，我第一次坐到电脑前的时候竟然慌了，面对VAX绿油油的屏幕不知道该做什么，好不容易敲入了一段程序，却满键盘找不到Ctrl键。这便是我第一次真正的编程体验，闹出了不少笑话。

这种情况在大二时得到了改观，我慢慢找到了编程的感觉，成绩也得到突飞猛进的提升。随后我去教授的实验室实习，接触到一个类似于现在的物联网项目，将很多电力设备，通过这个项目实现远程控制。这段时间，我体会到了做程序员的自豪感。

大学是我最快乐的一段时光，现在想来依然历历在目，但这种快乐与编程有关。如果说晚自习抢位子、卧谈会指点江山美女还是普遍现象的话，那么献血后冒雨踢足球、熄灯后到走廊偷电看世界杯而第二天是期末考试答疑，也就是那时的青春无敌才能驱使我们做出这些几近疯狂的事情。

谈到收获，感觉有四点：开阔了眼界、强健了体魄、确定了程序员这个职业、实现了财务独立。特别要说的是正是靠着拿英特尔的奖学金，我从大二开始就实现了财务独立，没再向父母要过钱，所以这对我后来加入英特尔也产生了间接的影响。

耕耘在英特尔

《程序员》：毕业后找工作的历程怎样？是一开始就选择了英特尔吗？

吴甘沙：我选工作的前提是工作地点要在北京。当时英特尔中国研究院也到上海来招聘了，但在招聘会上面试官纯正的美式英语，听得我糊里糊涂的，而且他讲的都是语音识别、人机交互、自然语言理解等方面的内容，和我的领域很不相同，于是我头都没回就走了。

后来，我应聘了另外一家跨国公司，它的实验室也是蛮有诱惑力的，拥有多名诺贝尔奖获得者，也发明了很多改变计算机领域的技术，于是我就去了，而且拿到了Offer。


吴甘沙现在努力帮助同事成长为下一个首席工程师

其实我是非常喜欢英特尔的，一方面是我拿了英特尔奖学金，另一方面是英特尔为奖学金获得者配备了导师以帮助我们成长。后来我有一次到北京开会，我在英特尔的导师软磨硬泡让我再去面试一次，加之这次面试又有了一个意外收获——我之所以要从上海转到北京工作，主要也是考虑到女朋友在北京读书，而英特尔中国研究院办公室就在我女朋友学校的对面，一想到以后可以去她学校食堂里面吃饭了——我的积极性一下子提了上来。结果这次面试非常顺利，英特尔，我来了。

《程序员》：刚加入英特尔时，感觉如何？

吴甘沙：我刚开始是在人机界面组，尽管不是我的“菜”，但靠着“只要你是认真地对待，一切弯路都是直路”的信念，我坚持了下来，并拿到了人生第一个专利。但在这个组我也生了一场大病：因为我当时的小老板每天下午3点来上班，凌晨3点下班，我想我刚进来总不好意思像他一样，所以我就每天早晨7~8点来，凌晨3点走，就这大概支撑了1~2个月……在生病的那段时间，我就在思考眼前的这条路是不是要继续走下去，我的梦想到底是什么，强项在哪里，什么样的工作能够体现我的优势，公司真正需要什么样的项目……恰好那时方之熙院长给中国研究院做了一个讲座，深深打动了我，这也成为一个催化剂，让我感到方院长讲的正是我热爱的方向，我一定要去那个组。接下来，随着这个组——编程系统组——顺理成章地成立，我便在第二年就申请调过去了。其实我是随遇而安的一个人，从2001年到现在一直在编程系统组。

《程序员》：如果从您加入英特尔到现在，划分几个阶段，您会怎样分？

吴甘沙：在这方面，我自己有一个观察，叫做四年周期论，即每四年换一个方向：2001年~2005年是移动受控运行时方向，2005年~2009年是并行编程和众核架构方向，2009年到现在是嵌入式系统软件方向，当然这次我希望能够待时间长一些，因为现在中国嵌入式市场的机会非常大，所以希望这次能够把四年周期论打破。

当然，四年周期论确实是非常巧合，如果非要划定几个点的话，我觉得有四点。

2004年是我职业生涯的第一个转折点，从一个技术人员变成了一个经理，从一个个人贡献者变成了一个团队管理者。实现这个转变要感谢我当时的经理，他事实上是在中国第一个获得首席工程师的人，但他属于美国编制，临时派过来帮我们。在他身上，我学到很多，无论是技术、研究方法论，还是管理。

2005年是第二个转折点，当时我们在移动的Java执行环境中，已经做到最快，便开始考虑下一阶段的技术发展。我们发现Java与操作系统有很多重叠的部分，比如内存管理、线程管理，而且能够给操作系统带来一些新东西，比如类型安全、受控执行，能够给整个的系统带来安全性的保证。因此，我们当时就考虑，怎样能够让Java支持多任务、让Java直接跑在硬

件上面、让Java与现存的操作系统相得益彰，其实在很多方面它已经接近Android的思路。但很不幸的是2005年整个公司在转型，右转弯到多核、众核，所以当时我们经历了非常痛苦的转变。这次转折，让我学到了应该怎样改变，同时，也学到什么东西应该坚持。实际上，2005年~2009年做并行编程的很多技术积累就是从前面这四年的周期里获得的。

2009年我转到了嵌入式系统软件方向，2011年获得首席工程师，这也算得上是转折。

“和而不同”的管理理念

《程序员》：在从技术个体转变成为管理者的过程中，您有没有做一些储备？请给处在这个转变过程中的人一些建议。

吴甘沙：储备的话倒谈不上。其实我的转变过程还是相当痛苦的，在很长一段时间里，老板给我的评价就是“你在抢你组员的活”，就是说尽管变成了经理，但仍放不下技术活，每次感觉进展有点缓慢，都亲自来做。所以这是一个漫长的过程。

整体来看，我是从三方面去提高：第一是自己去学，多看书，多参加培训；第二是向别人学，通过社交形成互动；第三是在工作中学。这些肯定不是一蹴而就的，需要时间的历练，慢慢地从与组员竞争的角色，转变成支持组员的角色。经理更多的是一种支持的角色，需要有公仆意识，为组员提供服务。

每个人的道路都不一样，一定会根据其个性、强项、弱点、所处的环境来找到一个最适合的一种管理理念和风格。

《程序员》：那么您的管理风格是什么？

吴甘沙：我属于那种润物细无声的人，因此不会像疾风骤雨那样，要求你做什么、批评你这样做不好，更多的是希望自己能够做好榜样，慢慢影响你。其实我整体的理念就是孔夫子的一句话——“和而不同”。

和就是和谐，我希望整个团队有一种非常好的氛围，能够产生化学反应。我的第一个经理给我的一个精神遗产就是：“你不一定选最好的人，但你要选对人，最适合这个团队、能够发生化学作用的。”

在和谐的基础上，我鼓励竞争，鼓励“不同”，鼓励独立的思考，鼓励争论，当然争论不应该是歇斯底里或者个人攻击。这个理念我秉承了好多年，而想法的来源就是文艺复兴中的“美第奇效应”，即不同的人，雕塑家、科学家、工程师、画家、音乐家凑到一起，才能产生文化的复兴。乔布斯在谈及他第一款产品之所以成功的原因时，便认为是他组织了一个团队，里面有历史学家、音乐家甚至动物学家，同时他们又是很好的工程师，能够创造出创新的产品。所以我特别鼓励员工独立思考，能够激发出不同的东西。

在实际操作中，一方面，我希望形成这种文化，让大家有创新的内在的驱动力；另一方面，我也会制造一种紧迫感，让大家觉得我们必须得创新了。在这种文化形成以后，我就放权，让每个人都有足够的空间和时间去在风险可控的前提下进行创新。我认真做好服务者的角色，为他们提供资源和支持，然后在他们做成以后，我会帮他们去推广。最终有了成绩以后，我帮他们申请奖励。

《程序员》：国家现在对物联网非常重视，嵌入式前景会很不错，您作为嵌入式领域的专家，请描绘一下将来会是什么景象？

吴甘沙：我们现在从E时代进入了U时代，U就是Ubiquitous（普遍存在的），U时代的一个特征就是计算设备的极大丰富，可能会到千亿级或者万亿级，每一种计算设备可能非常小，是深度嵌入。处在一种看不见嵌入设备的计算环境中，但同时它们确实在计算。

前两次产业革命，一次是PC，一次是互联网，中国只是一个旁观者、参与者。那么在物联网、三网融合充斥的U时代，我们国家一定是希望能够是成为一个产业的领导者。所以这对做嵌入式研究的人来说，是一个很大的机会。

《程序员》：请您对《程序员》杂志的读者说几句话。

吴甘沙：在IT人士当中，程序员是一个很难得的一个团体，就像刚才说的蜗牛一步一步地往上爬，所以希望《程序员》杂志能够为我们程序员提供心灵上的寄托和精神上的指引，共同进步。

Scrum的故事

文 / 司斌

2001年2月，17位敏捷先驱齐聚犹他雪鸟度假村，起草《敏捷宣言》的时候，Scrum只是众多方法中不太起眼的一个。十年之后，Scrum却成为最流行的敏捷方法，几乎成为敏捷的代名词。

本期名人堂人物就是Scrum的两位创始人——Jeff Sutherland与Ken Schwaber。

大家可能不会想到，Jeff Sutherland的第一份工作居然是美国空军战斗机飞行员，还曾于1967年获得了“壮志凌云”称号，完成过100次飞越北部越南的作战任务。服役后期，他到斯坦福大学拿下统计学硕士学位，并在美国空军学院教授数学统计学和概率学。11年军旅生涯结束后，他成为了科罗拉多医学院的教师并获得了博士学位。在诺贝尔化学奖得主莱纳斯·鲍林的赞助下，他以放射学、生物学及预防医学助理教授的身份参与了维生素与癌症研究中心的创立，担任八年国家癌症中心的主要研究员，负责科罗拉多地区所有癌症患者的数据统计和IT方案与研究，整合了国家注册、临床试验、流行病学研究和癌变的超级计算机数学模型。1983年，他进入了一家遍及北美、经营着150家银行的公司，职务为先进系统副总裁及ATM业务部总经理。此后，Sutherland先后担任了11家软件公司的CEO、CTO或者工程副总裁，积累了丰富的软件开发经验。

Scrum的另一位主角Ken Schwaber最初的职业也很特别——商船经理。在随后40多年开发生涯的前10年中，他曾经编写过操作系统，搞过嵌入式，为IBM大型机开发系统软件；先后在芝加哥大学、伊利诺伊理工学院、王安公司实验室工作，并逐渐展现出在软件开发方法上的天赋。在CASE工具和结构化方法热门的时候，他自己创办了ADM公司，从事软件开发方法培训服务。期间，公司开发了软件方法自动化工具MATE，用来生成各种软件流程所需的模板、计划等，生意很好。

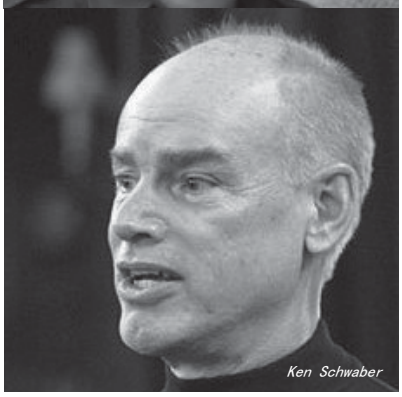
Sutherland和Schwaber相识于1980年代早期。1987年，两人开始合作。一天，Sutherland问Schwaber：“你们开发MATE工具都用了现在流行的哪一种方法？”“当然什么都没用，”Schwaber回答，“要不然公司早就完蛋了。”他们意识到问题的严重性，开始与开发者交谈，研究新方法。

1993年，Sutherland读到了两位日本管理教授竹内弘高和野中郁次郎介绍制造业里出现的新的产品开发方法Rugby（橄榄球）的文章。这种方法的特点是整个流程都由一个高性能、跨功能的团队执行到底。他受到启发，结合自己多年的经验，与Easel公司的John Scumniotales和Jeff McKenna一起开发了一套方法，取名为Scrum（来源于橄榄球术语，不是缩写）。

而Schwaber则从杜邦公司一位化工过程控制专家那里取经，意识到项目分为两种：确定性项目，一切都已经确定，可以自动化生产流程；实验性项目，充满不确定性，哪怕一点微小的变化也会牵一发而动全身，因此只能用各种仪表不断监控，随时做出调整——这就是每日站会的由来。

两人在一个IBM项目合作，并做了更详尽的研究，Scrum诞生了。1995年OOPSLA大会上他们第一次向世人介绍了Scrum。可当时，两个人的公司都还在做千年虫和各种重型开发方法咨询方面的业务呢。


Jeff Sutherland


Ken Schwaber

进入新世纪，互联网带来的巨变使敏捷方法受到了更多开发团队的青睐，而其中Scrum以其扩展性、门槛低、名字和术语更容易被项目经理接受等因素，逐渐成为最受欢迎的敏捷流派。而推出CSM等系列认证，虽然争议颇大，但客观上对Scrum扩大影响力起到了重要作用。

今天，Scrum的影响已经远远超出软件开发，成为零售、军事、风险投资甚至学校里完成各种任务的创新方法，正在改变着世界。著名思想家Steve Denning曾表示，如果有诺贝尔管理学奖的话，应该授予Scrum的创始人。

身处一个项目中，你必须具备足够的智慧，明白人们真正在说的是什么……

根据《项目百态》一书中相关章节改编

作者:
西乔

设计师，项目经理。06年起携创业团队从事Web技术外包开发及产品咨询顾问。

如果你有什么好玩的关于程序员的故事、对话、代码，愿意通过漫画的形式分享，请给西乔发邮件：arthur369@gmail.com